Mosaic Alpha Smart Contract Analysis

Independent Expert Assessment

Last modified: September 15, 2023.

1 EXECUTIVE SUMMARY

A team of the Critical Systems Research Group[1] at BME performed a smart contract source code audit for the Mosaic Alpha platform, on commission from DLabs Kft, using the Solidity source code provided by DLabs.

The key goal of the audit was to investigate whether the code base is free of known types of vulnerabilities and to show that known code weaknesses, if present, do not lead to exploitable vulnerabilities. To this end, we performed a manual code review and a case-by-case analysis of the issues reported for the code base by well-known static code analysis tools.

We believe that the code base is production-ready to the extent that the scope of our audit can reliably determine it, and assuming the suggested mitigations are performed. However, a detailed specification was not available for the code base. Such detailed documentation could provide an opportunity for deeper, semantic analyses of the behavior. As part of our work, we created a (back-engineered) initial structural documentation of the code base.

Moreover, the project employs a few potentially (but not necessarily) vulnerable solutions to fit the expected computational constraints of Ethereum execution environments. We found and described four important issues, for which we also report on the possible mitigations.

It is noted that manual audit and static analysis have well-known limitations in identifying vulnerabilities and that our analysis did not target the crypto-economics of Mosaic Alpha (e.g., fairness or stability properties). More formal investigations, specifically a requirement-based formal verification of contract logic and model-based crypto-economic analysis, can further increase confidence in the platform in the future.

[1]*The Critical Systems Research Group (formerly Fault Tolerant Systems Research Group) is a division of the Dept. of Measurement and Information Systems at the Budapest University of Technology and Economics, Budapest, Hungary. The analysis was conducted by Mihály Dobos-Kovács (code verification expert), Milán Mondok (code verification expert), Dr. Attila Klenik (technical lead and blockchain expert) and Dr. Imre Kocsis (PI).*

September 15, 2023, Budapest, Hungary

2 TABLE OF CONTENTS

3 THE MOSAIC ALPHA PLATFORM

The Mosaic Alpha platform aims to connect experienced, professional traders with novice investors who do not have a comprehensive knowledge of the crypto market. Traders can create ETF-like token baskets (containing two or more cryptocurrencies), which can be traded on the platform. Basket managers can earn commissions based on their baskets' performance. Investors can buy into baskets either by providing the contained tokens of the baskets or by providing USDT and involving the Pancake swap. The platform deducts three types of fees to fund the cost of operation and commissions, which are implemented through inflation. The platform also implements a complex two-level affiliate system, which incentivizes not only the investors who register through an affiliate link and the influencer who invited them but also users who invite influencers. Users can gain rewards (fee discounts, more commission, etc.) by staking the platform's utility token (KDX). A curated glossary of high-level terms is referred through Section 9 (Appendix D).

3.1 TARGET CODE BASE

We conducted the analysis on the files in the `contracts/mosaic-alpha-contracts` folder of the code base. Section 6.1 (Appendix A) provides a listing with file checksums.

Figure 1 provides an overview of the cross-dependencies of the key smart contracts.

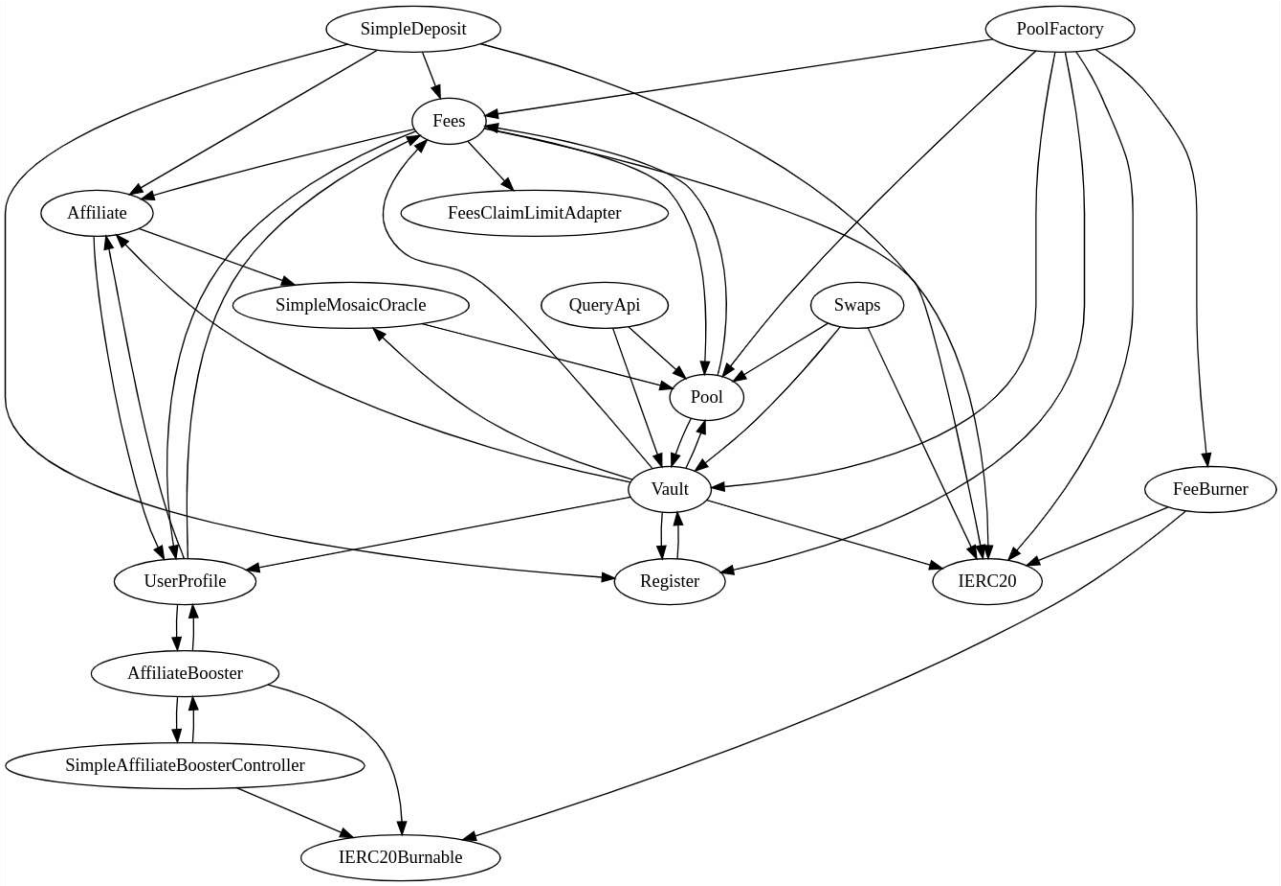

Figure 1. Cross-contract dependencies

3.2 COMPILING AND TESTING THE CODE BASE

The Mosaic Alpha platform was developed using the Hardhat[1] Ethereum development framework. The code needs NodeJS v16.17.1 to compile. The code is compiled, tested and deployed using Hardhat.

3.3 CODE BASE ASSESSMENT

The solution is implemented using 76 contracts (not counting unused contract code), containing altogether more than 10 kLOC. It is recognized that maintaining high code quality at such a scale in the Ethereum ecosystem poses inherent challenges. Nonetheless, as part of the audit process, it has become evident that there are specific areas warranting further attention with respect to code quality in a general sense.

A key concern is the absence of comprehensive system specification and documentation. This deficiency introduces significant complexities for anyone seeking to comprehend the nuances of the codebase. To this end, and to enable long-term maintainability and accessibility, we suggest the creation of a comprehensive documentation and rigorously commenting the source code.

It should be noted that the availability of good practice coding standards is only partial in this space and optimization concerns limit their applicability for this specific project; that said, the most important code weaknesses are checked for by static analysis tools.

4 AUDIT PROCESS

Our audit followed the process below.

1. Preparation: consultation with DLabs on the cryptoeconomic model and the implementation, reading and understanding the Solidity source code, build and testing environment setup
2. Manual audit of the usage of the Transparent Proxy pattern
3. Simplifying the implementation to facilitate the use of code analysis tooling
4. Static code analysis, evaluation of the results (assisted with call graph visualization techniques)
5. Checking the consistent application of authorization
6. Preliminary overflow analysis of critical math formulas

During the audit, the team consulted with DLabs multiple times with respect to code intent and the issues found.

4.1 MANUAL AUDIT OF THE USAGE OF THE TRANSPARENT PROXY PATTERN

The `Fees` and `Vault` contracts exceed the 24KB code size limit[2] of the Ethereum contracts. To circumvent this, Mosaic uses the Transparent Proxy pattern: splits these contracts into multiple `.sol` files and redirects calls when needed with the `delegatecall` operation.

The implementation of this pattern in the Mosaic platform is identical in its logic (apart from minor differences like storing the return value of the `returndatasize` call in a local variable instead of

[1] https://hardhat.org/
[2] https://eips.ethereum.org/EIPS/eip-170

passing it directly to the `returndata` function) to the `Proxy` implementation of the OpenZeppelin library[3].

Using the library implementation instead of reimplementing the logic would be preferable.

4.2 SIMPLIFYING THE TRANSPARENT PROXY PATTERN

After ensuring that the transparent proxy pattern does not affect validity, we merged the partitioned contract parts into a single contract to aid the static analysis tools.

4.3 REMOVING UNUSED CONTRACTS

The codebase of the Mosaic platform contains some unused contracts (see Section 6.2 in Appendix A for the complete list) that are not deployed. We removed these contracts to aid the static analysis tools.

4.4 APPLYING STATIC CODE ANALYSIS AND VISUALIZATIONS TECHNIQUES

During this review, we opted to use static source code analysis tools, and performed a manual evaluation of the issues raised by the tools. We also used visualization tools to aid with the manual review process. The following tools were used.

Slither: Slither[4] is a Solidity static analysis framework written in Python3. It runs a suite of vulnerability detectors, prints visual information about contract details, and provides an API to easily write custom analyses. Slither enables developers to find vulnerabilities, enhance code comprehension, and quickly prototype custom analyses.

Slither provided valuable output and most of our analysis is based on it.

Securify: Securify 2.0[5] is a security scanner for Ethereum smart contracts supported by the Ethereum Foundation and ChainSecurity. The core research behind Securify was conducted at the Secure, Reliable, and Intelligent Systems Lab at ETH Zurich.

Unfortunately, Securify is not maintained actively anymore, and only supports code up to Solidity version 0.6.12. Since most of the code base is based on 0.8.x, only a fraction of the code base (mainly the helpers) was analyzed with Securify.

Surya: Surya[6] is a utility tool for smart contract systems. It provides several visual outputs and information about the contracts' structure. Also supports querying the function call graph in multiple ways to aid in the manual inspection of contracts.

Surya provided multiple useful analyses that we used during this review. However, Surya did not handle the source code's inter-contract calling convention, so we modified Surya's source code to introduce support for it.

---

[3] https://github.com/OpenZeppelin/openzeppelin-contracts/blob/v4.8.2/contracts/proxy/Proxy.sol
[4] https://github.com/crytic/slither version 0.8.3
[5] https://github.com/eth-sri/securify2 commit 7f5d389a1ed3658538982e2b796824ba7c3ce013
[6] https://github.com/ConsenSys/surya version 0.4.6

After executing the aforementioned tools, we collected their outputs and prioritized the findings based on severity. We only investigated in detail the potential issues with a medium or high severity; these underwent manual inspection. The detailed report of the inspection is referred through Section 7 (Appendix B).

4.5 ANALYZING POTENTIAL REENTRANCY VULNERABILITIES

Special care was devoted to potential reentrancy-related vulnerabilities raised by Slither. To determine the possibility of reentrancy, first, we examined the potential modifiers on the code, as Slither disregards them during analysis. We found that in numerous cases, the code was protected by OpenZeppelin's `ReentrancyGuard`[7].

When this was not the case, we explored the call graph from the entry points indicated by the tool. We used Surya to generate the call graph and manually followed the chain of calls across multiple contracts. All in all, we concluded that in all cases,

1. either the reentrancy issue raised by Slither was actually protected by `ReentrancyGuard` or
2. the call sequence did not leave (and does not seem to be able to leave) the smart contract set of the Mosaic platform.

The detailed report of the inspection is referred through Section 10 (Appendix E).

4.6 CHECKING THE CONSISTENT APPLICATION OF AUTHORIZATION

We manually inspected the consistent use of authorization checks in the code base. The basis of the analysis was the report generated by Surya's `mdreport`. We focused on the public and external calls that modify the state of the smart contracts.

We manually inspected each such function, focusing on modifiers, authorization hidden in internal calls, or inside the function itself. The detailed report of the inspection is referred through Section 8 (Appendix C).

4.7 PRELIMINARY OVERFLOW ANALYSIS OF CRITICAL MATH FORMULAS

We performed a preliminary analysis of the mathematical calculations of the `Fees` contract focusing on whether the truncating cast operations can lead to loss of information. During the analysis, we only considered the syntactic form of the expressions, we did not take into consideration the semantic meaning of the individual variables.

During the analysis, we converted each expression to an SMT formula using the FixedSizeBitvectors[8] theory. Then we used the SMT solver Z3 to decide the satisfiability of the formula.

We concluded that overflow or data loss by truncation is possible in almost all calculations based on syntax alone. However, a manual review of the computations supports the hypothesis that it is unlikely that the necessary corner cases would be activated (requires unreasonably large balances).

---

[7] https://docs.openzeppelin.com/contracts/4.x/api/security#ReentrancyGuard

[8] https://smtlib.cs.uiowa.edu/theories-FixedSizeBitVectors.shtml

To further reason about the feasibility of such data loss, we would need to take into consideration the exact semantics and exact domain of the variables in the calculations and perform a symbolic execution style analysis (predicate abstraction and effect propagation), which was not done during this review.

As an alternative to proving freedom from over- and underflow situations, we suggest a systematic use of pre- and postconditions for the critical code regions (e.g., using assertions).

5 ISSUES FOUND AND REMEDIATIONS

During the audit, we found four major issues. These all have possible mitigations, which we also outline.

5.1 ISSUE #1

Issue: In the `contracts/mosaic-alpha-contracts/Vault/Swaps.sol` contract, the functions `addTokens` and `removeTokens` (on lines 29-49, see Listing 1) are missing proper authorization of the caller.

```
function addTokens(address[] calldata _tokens) external {
    for (uint i; i < _tokens.length; i++) {
        address token = _tokens[i];
        if (!tokens[token]) {
            _approveToken(token, type(uint256).max);
            tokens[token] = true;
            emit tokenAdded(token);
        }
    }
}

function removeTokens(address[] calldata _tokens) external {
    for (uint i; i < _tokens.length; i++) {
        address token = _tokens[i];
        if (tokens[token]) {
            _approveToken(token, 0);
            tokens[token] = false;
            emit tokenRemoved(token);
        }
    }
}
```

Listing 1: Source code context of Issue #1

Consequences: As there is no authorization, malicious actors can call these functions and add or remove tokens from the approved token list of the swap at will. However, this causes only temporary inconvenience for the swap since tokens can be re-approved.

Remediation: Update the contract.

5.2 Issue #2

Issue: The issue was discovered in the `_onBeforeBalanceTransfer` function of the `contracts/mosaic-alpha-contracts/Vault/Vault.sol` contract. The issue lies in the call to `IPool`'s `emitTransfer` function, where the third parameter should be `expAmount` instead of `amount (on lines 542-559, see Listing 2)`.

```
if (expAmount > 0) {
   address trader = IPool(_pool).creator();
   payload.feesContractBalanceBefore = getInternalBalance(fees,_pool).toUint128();
   IFees(fees).onBeforeTransfer(_pool, _from, _to, _internalBalance[_from][_pool]
                                , _internalBalance[_to][_pool]
                                , amount, originalTotalsupplyBase, trader
                                , payload
                                );
   // do the mint for the Trader
   poolStates[_pool].totalSupplyBase.amount += uint224(expAmount);
   poolStates[_pool].totalSupplyBase.timestamp = (block.timestamp).toUint32();
   emit TotalSupplyBaseChanged(_pool, poolStates[_pool].totalSupplyBase);
   _internalBalance[fees][_pool] += expAmount;
   _handlerForPoolOfUser(fees, _pool, true);
   // TODO casting!!!
   emit InternalBalanceChanged(fees, _pool, int256(uint256(expAmount)));
   if (true) IPool(_pool).emitTransfer(address(0), fees, amount);
   return;
}
```

Listing 2: Source code context for Issue #2

Consequences: Indexing services – that rely solely on the emitted event data – will display incorrect balances for the pool. However, the actual balances remain correct and can be queried explicitly.

Remediation: Cannot be mitigated since this contract cannot be updated without major redeployment; however, clearly communicating that the pool balances should not be determined based on the events should be a sufficient mitigation.

5.3 Issue #3

Issue: The `updatePerfFeePricePerLp` function of the `contracts/mosaic-alpha-contracts/Pool/Pool.sol` contract lacks authorization (on lines 810-822, see Listing 3).

```
function updatePerfFeePricePerLp(IVault.TotalSupplyBase calldata ts, uint256 valuePerLp)
  external returns (uint128) {
    return _updatePerfFeePricePerLp(ts, valuePerLp);
}

function _updatePerfFeePricePerLp(IVault.TotalSupplyBase calldata ts, uint256 valuePerLp)
  private returns (uint128 feeAmount) {
```

```
    // console.log("poolPf value: %s, valueperlp: %s lastvplp: %s", value, valuePerLp,
    // lastValuePerLp);
    if (lastValuePerLp == 0) {
        lastValuePerLp = (valuePerLp * uint256(105) / 100).toUint128();
    } else if (lastValuePerLp < valuePerLp) {
        feeAmount = (ts.amount * (valuePerLp - lastValuePerLp)
          * uint256(poolFees.performanceFee) / lastValuePerLp / 10000).toUint128();
        lastValuePerLp = valuePerLp.toUint128();
    }
}
```

Listing 3: Source code context for Issue #3

Consequences: The current implementation allows malicious actors to call the function and change the performance fee amount paid out to the managers of a pool. However, the amount range is constrained by the governance policy; thus only inconvenient fee decreases are possible.

Remediation: The `Pool` contract can be upgraded so that subsequent pool instances will use a correct implementation as prototype. However, already instantiated pools cannot be updated to fix the issue. Although this is a governance-level solution, these can be "phased out" without major inconvenience.

5.4 Issue #4

Issue: The implementation of `getTicketPrice` in the contracts/mosaic-alpha-`contracts/User/SimpleAffiliateBoosterController.sol` contract contains an implementation error – the return value is not assigned properly (on lines 63-66, see Listing 4).

```
// hook to get the current price of one ticket
function getTicketPrice(address user, uint256 count) external returns (uint256 price) {
    ticketPrice = price;
}
```

Listing 4: Source code context for Issue #4

Consequences: None. The contract is not deployed. It is only an example implementation to follow for other custom affiliate booster controller implementations.

Remediation: Not strictly needed; updating the example implementation is certainly advisable.

6 Appendix A - Artifacts

6.1 Target code base

Filename	Md5 checksum
./Deposit/Deposit.sol.disabled	0b11fdaad48f5badb3b42bd424cb0cf8
./Deposit/SimpleDeposit.sol	f63b603fb9de26a4d0072a6323c2cc7b
./Fees/FeeBurner.sol	208f59e5d3f69f15b2f5499649688a88
./Fees/Fees.sol	0caa6febf75af1072d87bcea84cd8224

./Fees/Fees2.sol.disabled	396773da315f4b1d306cf37a17d295b1
./Fees/FeesDelegated.sol	0216a10273b91aaf915efcbeea3a20bc
./Fees/MosaicFees.sol.disabled	588d6e7ad7b384aaee11cb9b5483dd46
./Fees/PoolFees.sol.disabled	623e5084dee0c15d425f77cd53f33577
./Governance/Governance.sol	975ec11b7dfdfc97b638543181c87528
./Governance/Governed.sol	b30bb044e08b34b134a75fcc294536b8
./helpers/Arrays.sol	901674deda1b97d0f172d79c30491899
./helpers/Controllable.sol	07b23a23cf85766e21cd4e4a1bb67394
./helpers/IEntropy.sol	9ac5f14f29547d82bebe10f62e691dde
./helpers/IERC20Burnable.sol	b9eac471ce627770d699ccd4908023dd
./helpers/ISystemRole.sol	b6ecfbad9b4d1923fb02fc263457a50b
./helpers/ISystemWhitelist.sol	50819d448e892bfc71738544e3eeccc2
./helpers/Ownable.sol	b393ab3e7b007bf684c44764521a903f
./helpers/pancake/interfaces/IERC20.sol	39b7f9523971270a05d8c20aaf1a9352
./helpers/pancake/interfaces/IPancakeCallee.sol	13aa5c53a6bd1f667a3d0bd2dcfcfb62
./helpers/pancake/interfaces/IPancakeERC20.sol	cc293e7b2057cb2a4db540534262bfad
./helpers/pancake/interfaces/IPancakeFactory.sol	717d7b34c24cd2d9614653c69749daae
./helpers/pancake/interfaces/IPancakeOracle.sol	38ff0735e3a711d8a67cbe11859341d6
./helpers/pancake/interfaces/IPancakePair.sol	642d06a561898083b851a04afb3e2d21
./helpers/pancake/interfaces/IPancakeRouter01.sol	56e6915f7873e2516723c74c5ac8b085
./helpers/pancake/interfaces/IPancakeRouter02.sol	f37b4aa435629a6469776209119f0247
./helpers/pancake/interfaces/IWETH.sol	61ca88bb5030a7f2b19c72c8862f8045
./helpers/pancake/libraries/Babylonian.sol	9145a4382f9a53eb268c1776b24f827a
./helpers/pancake/libraries/BitMath.sol	06025a99c2b890020bae8edc827e9656
./helpers/pancake/libraries/FixedPoint.sol	bc7888b2f2a998e217964ebf174b9083
./helpers/pancake/libraries/FullMath.sol	ee952b133c355986d1fc7b82e6366b7f
./helpers/pancake/libraries/Math.sol	0dffa3719a50992c6b20fd3f3de4fa00
./helpers/pancake/libraries/PancakeLibrary.sol	856e33045ebaa16e7327d0ebb208a8bf
./helpers/pancake/libraries/PancakeOracleLibrary.sol	40ecb7d6e2fc50f2604928b42d6b3a63
./helpers/pancake/libraries/SafeMath.sol	59dde9f75dc9ca8f5e3c04b654034fff
./helpers/pancake/libraries/TransferHelper.sol	a1223a3f75e25c152f78a5e19bc10832
./helpers/pancake/libraries/UQ112x112.sol	3cd0866c0b2f8fde2698272731337b2a
./helpers/pancake/PancakeERC20.sol	9ebdda43ba11fd7a0a17eb41e22f896c
./helpers/pancake/PancakeFactory.sol	85efb87519bdb234bbf584321d671209
./helpers/pancake/PancakePair.sol	d1ba0615b23539fcfab15a1fe9aeb3cd
./helpers/pancake/PancakeRouter.sol	3372057dc80404bcdb971a70a3043d13
./helpers/SimpleEntropy.sol	75740415f5c6e40f573235873a1893a0
./helpers/SystemRole.sol	1e82d70b274667a1a59bbf532fd55f8b
./helpers/SystemWhitelist.sol	12712f24994ac67d119338d45fb06c2f
./Interfaces/IAffiliate.sol	8f4e56679800096c94904c29dfa19e66
./Interfaces/IAffiliateBooster.sol	22323898b844206a3661f224693809ca

./Interfaces/IAffiliateBoosterController.sol	4cdf0a758b89efc8b7224c5078be6638
./Interfaces/IDeposit.sol	739e1c940913c7392feb07bb75abcf40
./Interfaces/IERC3156FlashBorrower.sol	982e6f7bf59f93ff5a3c285a85c9ba20
./Interfaces/IERC3156FlashLender.sol	72285586a34ae3b7f5e6bac8aa60fd66
./Interfaces/IFeeBurner.sol	cfb993ffa3108572c70a04aeb555b65e
./Interfaces/IFeeMapper.sol	f2dc91f793a8d427326248e032daec3f
./Interfaces/IFees.sol	0c416dc0b50b9215e995ea2b024e26be
./Interfaces/IFeesClaimLimitAdapter.sol	61952135ef5aff7d3ac4bbd2830f57a6
./Interfaces/IGovernance.sol	27117b17944788a851aff842be375a8f
./Interfaces/IGoverned.sol	77aa2ca74e174e1cfe93c32da38110b7
./Interfaces/IMosaicOracle.sol	ccb3cfd993ef7c233acb17c5649299ab
./Interfaces/IPool.sol	9b1517b6895da20969260c2e83600b81
./Interfaces/IQueryApi.sol	fa2d578400fff2dd0e89bf5326d2893c
./Interfaces/IRegister.sol	9a149aa3e0a74a8ac19aa291acb4d5fd
./Interfaces/ISimpleDeposit.sol	4f45ea144eb6211c2c4aef313eac3ab2
./Interfaces/ISwaps.sol	0e7c1d61113b6e1bd962d73c27a846b8
./Interfaces/IUserProfile.sol	dfb68379930a1be923e7ec6ccf358b4f
./Interfaces/IVault.sol	8f56c0557b571a0ab635d853c4b6463e
./Interfaces/IVaultDelegated.sol	abbabf291037d72a7b2e5f0eb5b56e40
./Libraries/PoolLibrary.sol	94606d1cf95d63ea6b5ae40d9e031de9
./Oracle/Initializable.sol	151b5b677b4fca274391a85eacb4400a
./Oracle/MosaicOracle.sol.bak	a1238703b97960dd7676fdedcea84f51
./Oracle/PancakeOracle.sol.bak	ccd205e23d7c4d4c16fe093dfead64c0
./Oracle/SimpleMosaicOracle.sol	bd27b6a75e43cdfb4799a9c1e4852fbc
./Pool/Math.sol	22a001f44b167fbc02f71a82c376c2ac
./Pool/Pool.sol	bf5e407362bdb8d8afbef33216eb7b22
./Pool/PoolFactory.sol	559e0ad61a4a1eca06cfef69ef655089
./Query/QueryApi.sol	20632d0dc5d3000e0ac6f3a6e5557a32
./Register/Register.sol	edaf5668b6ec1eeeff6af78fb0c766cf
./User/Affiliate.sol	d1dc837264dbcb5b400190271f11fb52
./User/Affiliate_old.sol	54160a09df08ac524e7395f83d8b40a8
./User/AffiliateBooster.sol	59768a5130e314c166db616b0fc60078
./User/SimpleAffiliateBoosterController.sol	c257dc667fd8b7113d422dc7b1989ee5
./User/UserProfile.sol	c7746c8e64a0962f90ab18cc1baf1452
./Vault/FlashLoans.sol	f2466bab1108438fcd9f554739a09f90
./Vault/Swaps.sol	666d14ab4a1ea2c5a44a9b04b332691a
./Vault/Vault.sol	672591e456b5149d17f9e59fc5b1530e
./Vault/VaultDelegated.sol	302d916918a5e403a893e34c05a5c598
./Vault/VaultExt.sol	a91d777a296775b10d5df9b6c7d1411a

6.2 UNUSED CONTRACTS REMOVED BEFORE ANALYSIS

We removed the following unused files before the analysis:

- contracts/mosaic-alpha-contracts
  - Deposit
    - Deposit.sol.disabled
  - Fees
    - Fees2.sol.disabled
    - MosaicFees.sol.disabled
    - PoolFees.sol.disabled
  - Interfaces
    - IDeposit.sol
    - IFeeMapper.sol
  - Oracle
    - MosaicOracle.sol.bak
    - PancakeOracle.sol.bak

6.3 MODIFICATION OF SURYA

The patch in Listing 5 applies the modifications we performed on commit 774b28e02e9088ec5395bc93ec8e37332a7ac116:

```
diff --git a/src/utils/parserHelpers.js b/src/utils/parserHelpers.js
index e1eda0d..4ebec14 100644
--- a/src/utils/parserHelpers.js
+++ b/src/utils/parserHelpers.js
@@ -41,7 +41,7 @@
const parserHelpers = {
     // @TODO: replace lowercase for better filtering
     return expr.type === 'FunctionCall'
         && expr.expression.hasOwnProperty('name')
-        && contractNames.includes(expr.expression.name[0]);
+        && contractNames.includes(expr.expression.name instanceof Array ?
expr.expression.name[0] : expr.expression.name);
},

   isUserDefinedDeclaration: node => {
```

Listing 5: Patch applied to Surya

7 APPENDIX B – SLITHER OUTPUT

The complete output is extracted into the following file attachment:

"Mosaic Alpha Smart Contract Analysis Appendix B – Slither Output.pdf" (SHA256: 3811356d0ef495fac08a9db1a5e964fd77d817dfe311bf157d052ea33b04567f)

8 APPENDIX C – AUTHORIZATION CHECK REPORT

The complete listing is extracted into the following file attachment:

“Mosaic Alpha Smart Contract Analysis Appendix C – Authorization Report.pdf” (SHA256: 777abdccc2e372b95eb9f2bf4590fb97c7d28af00e2a55c9bfb3f622aadf47bb)

9 APPENDIX D – GLOSSARY

A high-level glossary of the target domain is extracted into the following file attachment:

“Mosaic Alpha Smart Contract Analysis Appendix D – Glossary.pdf” (SHA256: 22234deb640d7dce9e6a5d5f71fa6db9696fe4ef91264fa8ca3eeb0b2d9fea16)

10 APPENDIX E – CONTRACT EXPLORATION

The documentation for contract behavior identification and review is extracted into the following file attachment:

“Mosaic Alpha Smart Contract Analysis Appendix E – Contract Exploration.pdf” (SHA256: f98fc169b7e2664ba0715786cf4494f2a3536ea35c94ec16dee73bb7d45124b2)

Slither Output

Mosaic Alpha Smart Contract Analysis Appendix B

Contents

Slither

Tool output

High impact

weak-PRNG

```
PancakePair._update(uint256,uint256,uint112,uint112) (contracts/mosaic-alpha-contracts/helpers/pancake/PancakePair.sol#75-88) uses a weak PRNG:
"blockTimestamp = uint32(block.timestamp % 2 ** 32) (contracts/mosaic-alpha-contracts/helpers/pancake/PancakePair.sol#77)"
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#weak-PRNG

AffiliateBooster.pickNextReferral() (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#372-410) uses a weak PRNG: "threshold = seed %
(state.totalPlayerWeight) (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#383)"
AffiliateBooster.pickNextReferral() (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#372-410) uses a weak PRNG: "pickedAddress =
epochTickets[state.currentEpoch - i][seed % epochTickets[state.currentEpoch - i].length] (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#389)"
AffiliateBooster.pickNextReferral() (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#372-410) uses a weak PRNG: "epochTickets[state.currentEpoch
- i][seed % epochTickets[state.currentEpoch - i].length] = epochTickets[state.currentEpoch - i][epochTickets[state.currentEpoch - i].length - 1]
(contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#390)"
PancakeOracleLibrary.currentBlockTimestamp() (contracts/mosaic-alpha-contracts/helpers/pancake/libraries/PancakeOracleLibrary.sol#11-13) uses a weak PRNG:
"uint32(block.timestamp % 2 ** 32) (contracts/mosaic-alpha-contracts/helpers/pancake/libraries/PancakeOracleLibrary.sol#12)"
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#weak-PRNG
```

name-reused

```
IVault is re-used:
    - IVault (contracts/mosaic-alpha-bot/Interfaces/Interfaces/IVault.sol#4-49)
    - IVault (contracts/mosaic-alpha-contracts/Interfaces/IVault.sol#7-301)
IQueryApi is re-used:
    - IQueryApi (contracts/mosaic-alpha-bot/Interfaces/Interfaces/IQueryApi.sol#5-29)
    - IQueryApi (contracts/mosaic-alpha-contracts/Interfaces/IQueryApi.sol#5-28)
Address is re-used:
    - Address (node_modules/@openzeppelin/contracts/utils/Address.sol#9-222)
    - Address (contracts/.deps/npm/@openzeppelin/contracts/utils/Address.sol#9-244)
Ownable is re-used:
    - Ownable (node_modules/@openzeppelin/contracts/access/Ownable.sol#20-83)
    - Ownable (contracts/mosaic-alpha-contracts/helpers/Ownable.sol#18-81)
IERC20 is re-used:
    - IERC20 (node_modules/@openzeppelin/contracts/token/ERC20/IERC20.sol#9-82)
    - IERC20 (contracts/.deps/npm/@openzeppelin/contracts/token/ERC20/IERC20.sol#9-82)
    - IERC20 (contracts/mosaic-alpha-bot/Interfaces/Interfaces/IERC20.sol#4-10)
IPool is re-used:
    - IPool (contracts/mosaic-alpha-bot/Interfaces/Interfaces/IPool.sol#4-8)
    - IPool (contracts/mosaic-alpha-contracts/Interfaces/IPool.sol#7-132)
FixedPoint is re-used:
    - FixedPoint (contracts/mosaic-alpha-contracts/Pool/Math.sol#4-609)
    - FixedPoint (contracts/mosaic-alpha-contracts/helpers/pancake/libraries/FixedPoint.sol#9-147)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#name-reused
```

reentrancy-vulnerabilities

```
Reentrancy in Vault.bid(uint256,uint256,bool,bool,uint256) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1163-1213):
    External calls:
    - _depositToInternalBalance(_auctionDetails.tokenToBuy,msg.sender,_auctionDetails.poolAddress,_toPay,false) (contracts/mosaic-alpha-
contracts/Vault/Vault.sol#1203)
        - returndata = address(token).functionCall(data,SafeERC20: low-level call failed)
(node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#110)
        - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#137)
        - IERC20(token).safeTransferFrom(from,address(this),amount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#463)
    - _withdrawFromInternalBalance(_auctionDetails.tokenToSell,_auctionDetails.poolAddress,msg.sender,amount) (contracts/mosaic-alpha-
contracts/Vault/Vault.sol#1204)
        - returndata = address(token).functionCall(data,SafeERC20: low-level call failed)
(node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#110)
        - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#137)
        - IERC20(token).safeTransfer(to,amount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#476)
    External calls sending eth:
    - _depositToInternalBalance(_auctionDetails.tokenToBuy,msg.sender,_auctionDetails.poolAddress,_toPay,false) (contracts/mosaic-alpha-
contracts/Vault/Vault.sol#1203)
        - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#137)
    - _withdrawFromInternalBalance(_auctionDetails.tokenToSell,_auctionDetails.poolAddress,msg.sender,amount) (contracts/mosaic-alpha-
contracts/Vault/Vault.sol#1204)
        - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#137)
    State variables written after the call(s):
    - _withdrawFromInternalBalance(_auctionDetails.tokenToSell,_auctionDetails.poolAddress,msg.sender,amount) (contracts/mosaic-alpha-
contracts/Vault/Vault.sol#1204)
        - _internalBalance[from][token] -= amount (contracts/mosaic-alpha-contracts/Vault/Vault.sol#475)
Reentrancy in Vault.bid(uint256,uint256,bool,bool,uint256) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1163-1213):
    External calls:
    - _transferInternalBalance(_auctionDetails.tokenToBuy,msg.sender,_auctionDetails.poolAddress,_toPay) (contracts/mosaic-alpha-
contracts/Vault/Vault.sol#1199)
        - IMosaicOracle(mosaicOracle).consultUsdPoolPricePerLp(poolAddr,ts) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#703)
        - IFees(fees).onBeforeTransfer(_pool,_from,_to,_internalBalance[_from][_pool],_internalBalance[_to]
[_pool],amount,originalTotalsupplyBase,trader,payload) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#598-601)
        - IPool(token).emitTransfer(from,to,amount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#499)
        - performanceFeesToMint = (IPool(_pool).updatePerfFeePricePerLp(tsTmp,valuePerLp)) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#568)
        - IPool(_pool).emitTransfer(address(0),fees,amount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#610)
        - IFees(fees).onBeforeTransfer(_pool,_from,_to,_internalBalance[_from][_pool],_internalBalance[_to][_pool],amount,0,address(0),payload)
(contracts/mosaic-alpha-contracts/Vault/Vault.sol#614)
    - _transferInternalBalance(_auctionDetails.tokenToSell,_auctionDetails.poolAddress,msg.sender,amount) (contracts/mosaic-alpha-
contracts/Vault/Vault.sol#1201)
        - IMosaicOracle(mosaicOracle).consultUsdPoolPricePerLp(poolAddr,ts) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#703)
        - IFees(fees).onBeforeTransfer(_pool,_from,_to,_internalBalance[_from][_pool],_internalBalance[_to]
[_pool],amount,originalTotalsupplyBase,trader,payload) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#598-601)
        - IPool(token).emitTransfer(from,to,amount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#499)
        - performanceFeesToMint = (IPool(_pool).updatePerfFeePricePerLp(tsTmp,valuePerLp)) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#568)
        - IPool(_pool).emitTransfer(address(0),fees,amount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#610)
        - IFees(fees).onBeforeTransfer(_pool,_from,_to,_internalBalance[_from][_pool],_internalBalance[_to][_pool],amount,0,address(0),payload)
(contracts/mosaic-alpha-contracts/Vault/Vault.sol#614)
    - _depositToInternalBalance(_auctionDetails.tokenToBuy,msg.sender,_auctionDetails.poolAddress,_toPay,false) (contracts/mosaic-alpha-
contracts/Vault/Vault.sol#1203)
        - returndata = address(token).functionCall(data,SafeERC20: low-level call failed)
(node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#110)
        - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#137)
        - IERC20(token).safeTransferFrom(from,address(this),amount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#463)
    - _withdrawFromInternalBalance(_auctionDetails.tokenToSell,_auctionDetails.poolAddress,msg.sender,amount) (contracts/mosaic-alpha-
contracts/Vault/Vault.sol#1204)
```

```
        - returndata = address(token).functionCall(data,SafeERC20: low-level call failed)
(node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#110)
        - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#137)
        - IERC20(token).safeTransfer(to,amount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#476)
    External calls sending eth:
    - _depositToInternalBalance(_auctionDetails.tokenToBuy,msg.sender,_auctionDetails.poolAddress,_toPay,false) (contracts/mosaic-alpha-
contracts/Vault/Vault.sol#1203)
        - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#137)
    - _withdrawFromInternalBalance(_auctionDetails.tokenToSell,_auctionDetails.poolAddress,msg.sender,amount) (contracts/mosaic-alpha-
contracts/Vault/Vault.sol#1204)
        - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#137)
    State variables written after the call(s):
    - auctions[auctionId].remainingAmount -= amount (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1208)
Reentrancy in Vault.swap(address,bool,address,address,uint256,bool,uint256,uint64) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#969-1032):
    External calls:
    - _depositToInternalBalance(tokenIn,msg.sender,poolAddress,_amountIn,true) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1017)
        - returndata = address(token).functionCall(data,SafeERC20: low-level call failed)
(node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#110)
        - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#137)
        - IERC20(token).safeTransferFrom(from,address(this),amount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#463)
    - _withdrawFromInternalBalance(tokenOut,poolAddress,msg.sender,_amountOut) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1019)
        - returndata = address(token).functionCall(data,SafeERC20: low-level call failed)
(node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#110)
        - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#137)
        - IERC20(token).safeTransfer(to,amount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#476)
    External calls sending eth:
    - _depositToInternalBalance(tokenIn,msg.sender,poolAddress,_amountIn,true) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1017)
        - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#137)
    - _withdrawFromInternalBalance(tokenOut,poolAddress,msg.sender,_amountOut) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1019)
        - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#137)
    State variables written after the call(s):
    - _withdrawFromInternalBalance(tokenOut,poolAddress,msg.sender,_amountOut) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1019)
        - _internalBalance[from][token] -= amount (contracts/mosaic-alpha-contracts/Vault/Vault.sol#475)
Reentrancy in Vault.swap(address,bool,address,address,uint256,bool,uint256,uint64) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#969-1032):
    External calls:
    - _transferInternalBalance(tokenIn,msg.sender,poolAddress,_amountIn) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1010)
        - IMosaicOracle(mosaicOracle).consultUsdPoolPricePerLp(poolAddr,ts) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#703)
        - IFees(fees).onBeforeTransfer(_pool,_from,_to,_internalBalance[_from][_pool],_internalBalance[_to]
[_pool],amount,originalTotalsupplyBase,trader,payload) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#598-601)
        - IPool(token).emitTransfer(from,to,amount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#499)
        - performanceFeesToMint = (IPool(_pool).updatePerfFeePricePerLp(tsTmp,valuePerLp)) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#568)
        - IPool(_pool).emitTransfer(address(0),fees,amount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#610)
        - IFees(fees).onBeforeTransfer(_pool,_from,_to,_internalBalance[_from][_pool],_internalBalance[_to][_pool],amount,0,address(0),payload)
(contracts/mosaic-alpha-contracts/Vault/Vault.sol#614)
    - _transferInternalBalance(tokenOut,poolAddress,msg.sender,_amountOut) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1012)
        - IMosaicOracle(mosaicOracle).consultUsdPoolPricePerLp(poolAddr,ts) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#703)
        - IFees(fees).onBeforeTransfer(_pool,_from,_to,_internalBalance[_from][_pool],_internalBalance[_to]
[_pool],amount,originalTotalsupplyBase,trader,payload) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#598-601)
        - IPool(token).emitTransfer(from,to,amount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#499)
        - performanceFeesToMint = (IPool(_pool).updatePerfFeePricePerLp(tsTmp,valuePerLp)) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#568)
        - IPool(_pool).emitTransfer(address(0),fees,amount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#610)
        - IFees(fees).onBeforeTransfer(_pool,_from,_to,_internalBalance[_from][_pool],_internalBalance[_to][_pool],amount,0,address(0),payload)
(contracts/mosaic-alpha-contracts/Vault/Vault.sol#614)
    - _depositToInternalBalance(tokenIn,msg.sender,poolAddress,_amountIn,true) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1017)
        - returndata = address(token).functionCall(data,SafeERC20: low-level call failed)
(node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#110)
        - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#137)
        - IERC20(token).safeTransferFrom(from,address(this),amount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#463)
    - _withdrawFromInternalBalance(tokenOut,poolAddress,msg.sender,_amountOut) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1019)
        - returndata = address(token).functionCall(data,SafeERC20: low-level call failed)
(node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#110)
        - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#137)
        - IERC20(token).safeTransfer(to,amount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#476)
    - _transferInternalBalance(tokenOut,poolAddress,vaultState.feeTo,_feeToProtocol) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1027)
        - IMosaicOracle(mosaicOracle).consultUsdPoolPricePerLp(poolAddr,ts) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#703)
        - IFees(fees).onBeforeTransfer(_pool,_from,_to,_internalBalance[_from][_pool],_internalBalance[_to]
[_pool],amount,originalTotalsupplyBase,trader,payload) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#598-601)
        - IPool(token).emitTransfer(from,to,amount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#499)
        - performanceFeesToMint = (IPool(_pool).updatePerfFeePricePerLp(tsTmp,valuePerLp)) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#568)
        - IPool(_pool).emitTransfer(address(0),fees,amount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#610)
        - IFees(fees).onBeforeTransfer(_pool,_from,_to,_internalBalance[_from][_pool],_internalBalance[_to][_pool],amount,0,address(0),payload)
(contracts/mosaic-alpha-contracts/Vault/Vault.sol#614)
    External calls sending eth:
    - _depositToInternalBalance(tokenIn,msg.sender,poolAddress,_amountIn,true) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1017)
        - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#137)
    - _withdrawFromInternalBalance(tokenOut,poolAddress,msg.sender,_amountOut) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1019)
        - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#137)
    State variables written after the call(s):
    - _transferInternalBalance(tokenOut,poolAddress,vaultState.feeTo,_feeToProtocol) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1027)
        - _internalBalance[from][token] -= amount (contracts/mosaic-alpha-contracts/Vault/Vault.sol#494)
        - _internalBalance[to][token] += amount (contracts/mosaic-alpha-contracts/Vault/Vault.sol#495)
        - _internalBalance[fees][_pool] += expAmount (contracts/mosaic-alpha-contracts/Vault/Vault.sol#606)
        - _internalBalance[from][token] -= amount (contracts/mosaic-alpha-contracts/Vault/Vault.sol#502)
        - _internalBalance[to][token] += amount (contracts/mosaic-alpha-contracts/Vault/Vault.sol#503)
    - _transferInternalBalance(tokenOut,poolAddress,vaultState.feeTo,_feeToProtocol) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1027)
        - poolOfUser[user].push(poolId) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1226)
        - poolOfUser[user][index_scope_0] = replacementPoolId (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1240)
        - poolOfUser[user].pop() (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1241)
    - _transferInternalBalance(tokenOut,poolAddress,vaultState.feeTo,_feeToProtocol) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1027)
        - poolOfUserIndexes[user][poolId] = (poolOfUser[user].length).toUint32() (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1228)
        - poolOfUserIndexes[user][replacementPoolId] = uint32(index_scope_0) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1239)
        - poolOfUserIndexes[user][poolId] = 0 (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1242)
    - _transferInternalBalance(tokenOut,poolAddress,vaultState.feeTo,_feeToProtocol) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1027)
        - poolStates[_pool].lastTrailingTimestamp = uint48(block.timestamp) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#560)
        - poolStates[_pool].totalSupplyBase.amount += uint224(expAmount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#603)
        - poolStates[_pool].totalSupplyBase.timestamp = (block.timestamp).toUint32() (contracts/mosaic-alpha-contracts/Vault/Vault.sol#604)
        - poolStates[_pool].lastPerformanceTimestamp = uint48(block.timestamp) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#570)
Reentrancy in Vault.swap(address,bool,address,address,uint256,bool,uint256,uint64) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#969-1032):
    External calls:
    - _transferInternalBalance(tokenIn,msg.sender,poolAddress,_amountIn) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1010)
        - IMosaicOracle(mosaicOracle).consultUsdPoolPricePerLp(poolAddr,ts) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#703)
        - IFees(fees).onBeforeTransfer(_pool,_from,_to,_internalBalance[_from][_pool],_internalBalance[_to]
[_pool],amount,originalTotalsupplyBase,trader,payload) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#598-601)
        - IPool(token).emitTransfer(from,to,amount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#499)
        - performanceFeesToMint = (IPool(_pool).updatePerfFeePricePerLp(tsTmp,valuePerLp)) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#568)
        - IPool(_pool).emitTransfer(address(0),fees,amount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#610)
        - IFees(fees).onBeforeTransfer(_pool,_from,_to,_internalBalance[_from][_pool],_internalBalance[_to][_pool],amount,0,address(0),payload)
(contracts/mosaic-alpha-contracts/Vault/Vault.sol#614)
    - _transferInternalBalance(tokenOut,poolAddress,msg.sender,_amountOut) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1012)
        - IMosaicOracle(mosaicOracle).consultUsdPoolPricePerLp(poolAddr,ts) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#703)
        - IFees(fees).onBeforeTransfer(_pool,_from,_to,_internalBalance[_from][_pool],_internalBalance[_to]
[_pool],amount,originalTotalsupplyBase,trader,payload) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#598-601)
        - IPool(token).emitTransfer(from,to,amount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#499)
        - performanceFeesToMint = (IPool(_pool).updatePerfFeePricePerLp(tsTmp,valuePerLp)) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#568)
        - IPool(_pool).emitTransfer(address(0),fees,amount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#610)
        - IFees(fees).onBeforeTransfer(_pool,_from,_to,_internalBalance[_from][_pool],_internalBalance[_to][_pool],amount,0,address(0),payload)
(contracts/mosaic-alpha-contracts/Vault/Vault.sol#614)
```

```
    - _depositToInternalBalance(tokenIn,msg.sender,poolAddress,_amountIn,true) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1017)
        - returndata = address(token).functionCall(data,SafeERC20: low-level call failed)
(node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#110)
        - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#137)
        - IERC20(token).safeTransferFrom(from,address(this),amount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#463)
    - _withdrawFromInternalBalance(tokenOut,poolAddress,msg.sender,_amountOut) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1019)
        - returndata = address(token).functionCall(data,SafeERC20: low-level call failed)
(node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#110)
        - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#137)
        - IERC20(token).safeTransfer(to,amount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#476)
    - _transferInternalBalance(tokenIn,poolAddress,vaultState.feeTo,_feeToProtocol) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1027)
        - IMosaicOracle(mosaicOracle).consultUsdPoolPricePerLp(poolAddr,ts) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#703)
        - IFees(fees).onBeforeTransfer(_pool,_from,_to,_internalBalance[_from][_pool],_internalBalance[_to]
[_pool],amount,originalTotalsupplyBase,trader,payload) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#598-601)
        - IPool(token).emitTransfer(from,to,amount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#499)
        - performanceFeesToMint = (IPool(_pool).updatePerfFeePricePerLp(tsTmp,valuePerLp)) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#568)
        - IPool(_pool).emitTransfer(address(0),fees,amount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#610)
        - IFees(fees).onBeforeTransfer(_pool,_from,_to,_internalBalance[_from][_pool],_internalBalance[_to][_pool],amount,0,address(0),payload)
(contracts/mosaic-alpha-contracts/Vault/Vault.sol#614)
    External calls sending eth:
    - _depositToInternalBalance(tokenIn,msg.sender,poolAddress,_amountIn,true) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1017)
        - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#137)
    - _withdrawFromInternalBalance(tokenOut,poolAddress,msg.sender,_amountOut) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1019)
        - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#137)
    State variables written after the call(s):
    - _transferInternalBalance(tokenIn,poolAddress,vaultState.feeTo,_feeToProtocol) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1027)
        - _internalBalance[from][token] -= amount (contracts/mosaic-alpha-contracts/Vault/Vault.sol#494)
        - _internalBalance[to][token] += amount (contracts/mosaic-alpha-contracts/Vault/Vault.sol#495)
        - _internalBalance[fees][_pool] += expAmount (contracts/mosaic-alpha-contracts/Vault/Vault.sol#606)
        - _internalBalance[from][token] -= amount (contracts/mosaic-alpha-contracts/Vault/Vault.sol#502)
        - _internalBalance[to][token] += amount (contracts/mosaic-alpha-contracts/Vault/Vault.sol#503)
    - _transferInternalBalance(tokenIn,poolAddress,vaultState.feeTo,_feeToProtocol) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1027)
        - poolOfUser[user].push(poolId) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1226)
        - poolOfUser[user][index_scope_0] = replacementPoolId (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1240)
        - poolOfUser[user].pop() (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1241)
    - _transferInternalBalance(tokenIn,poolAddress,vaultState.feeTo,_feeToProtocol) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1027)
        - poolOfUserIndexes[user][poolId] = (poolOfUser[user].length).toUint32() (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1228)
        - poolOfUserIndexes[user][replacementPoolId] = uint32(index_scope_0) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1239)
        - poolOfUserIndexes[user][poolId] = 0 (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1242)
    - _transferInternalBalance(tokenIn,poolAddress,vaultState.feeTo,_feeToProtocol) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1027)
        - poolStates[_pool].lastTrailingTimestamp = uint48(block.timestamp) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#560)
        - poolStates[_pool].totalSupplyBase.amount += uint224(expAmount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#603)
        - poolStates[_pool].totalSupplyBase.timestamp = (block.timestamp).toUint32() (contracts/mosaic-alpha-contracts/Vault/Vault.sol#604)
        - poolStates[_pool].lastPerformanceTimestamp = uint48(block.timestamp) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#570)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities
```

state-variable-shadowing

```
SimpleDeposit._owner (contracts/mosaic-alpha-contracts/Deposit/SimpleDeposit.sol#57) shadows:
    - SystemRole._owner (contracts/mosaic-alpha-contracts/helpers/SystemRole.sol#57)
    - Ownable._owner (node_modules/@openzeppelin/contracts/access/Ownable.sol#21)
SystemRole._owner (contracts/mosaic-alpha-contracts/helpers/SystemRole.sol#57) shadows:
    - Ownable._owner (node_modules/@openzeppelin/contracts/access/Ownable.sol#21)
SystemWhitelist._owner (contracts/mosaic-alpha-contracts/helpers/SystemWhitelist.sol#25) shadows:
    - SystemRole._owner (contracts/mosaic-alpha-contracts/helpers/SystemRole.sol#57)
    - Ownable._owner (node_modules/@openzeppelin/contracts/access/Ownable.sol#21)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variable-shadowing
```

unchecked-transfer

```
PancakeRouter.removeLiquidity(address,address,uint256,uint256,uint256,address,uint256) (contracts/mosaic-alpha-
contracts/helpers/pancake/PancakeRouter.sol#105-121) ignores return value by IPancakePair(pair).transferFrom(msg.sender,pair,liquidity) (contracts/mosaic-
alpha-contracts/helpers/pancake/PancakeRouter.sol#115)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-transfer

Vault.emergencyWithdrawAll(address,address,uint256) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1364-1368) ignores return value by
IERC20(_token).transfer(_to,_amount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1367)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-transfer

FeeBurner.burn() (contracts/mosaic-alpha-contracts/Fees/FeeBurner.sol#70-76) ignores return value by
IERC20(feeToken).transfer(feeTo,IERC20(feeToken).balanceOf(address(this))) (contracts/mosaic-alpha-contracts/Fees/FeeBurner.sol#74)
PoolFactory.create(string,string,uint8,address[],uint32[],uint256[],IFees.MosaicPoolFees,address,bool) (contracts/mosaic-alpha-
contracts/Pool/PoolFactory.sol#120-263) ignores return value by IERC20(tokens[i]).transferFrom(msg.sender,address(this),initialLiquidity[i])
(contracts/mosaic-alpha-contracts/Pool/PoolFactory.sol#196-200)
AffiliateBooster.purchaseTicket(address,uint256) (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#322-331) ignores return value by
IERC20Burnable(state.feeToken).transfer(state.feeTo,_c * state.ticketPrice) (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#328)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-transfer
```

Medium impact

misuse-of-a-boolean-constant

```
Vault._onBeforeBalanceTransfer(address,address,address,uint256,bool,bool,bool) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#583-615) uses a Boolean
constant improperly:
    -true (contracts/mosaic-alpha-contracts/Vault/Vault.sol#610)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#misuse-of-a-boolean-constant
```

divide-before-multiply

```
FixedPoint.pow(uint256,uint256) (contracts/mosaic-alpha-contracts/Pool/Math.sol#188-234) performs a multiplication on the result of a division:
    -logx_times_y = ((ln_36_x / ONE_18) * y_int256 + ((ln_36_x % ONE_18) * y_int256) / ONE_18) (contracts/mosaic-alpha-contracts/Pool/Math.sol#221)
FixedPoint.exp(int256) (contracts/mosaic-alpha-contracts/Pool/Math.sol#241-376) performs a multiplication on the result of a division:
    -product = (product * a2) / ONE_20 (contracts/mosaic-alpha-contracts/Pool/Math.sol#289)
    -product = (product * a3) / ONE_20 (contracts/mosaic-alpha-contracts/Pool/Math.sol#293)
FixedPoint.exp(int256) (contracts/mosaic-alpha-contracts/Pool/Math.sol#241-376) performs a multiplication on the result of a division:
    -product = (product * a3) / ONE_20 (contracts/mosaic-alpha-contracts/Pool/Math.sol#293)
    -product = (product * a4) / ONE_20 (contracts/mosaic-alpha-contracts/Pool/Math.sol#297)
FixedPoint.exp(int256) (contracts/mosaic-alpha-contracts/Pool/Math.sol#241-376) performs a multiplication on the result of a division:
    -product = (product * a4) / ONE_20 (contracts/mosaic-alpha-contracts/Pool/Math.sol#297)
    -product = (product * a5) / ONE_20 (contracts/mosaic-alpha-contracts/Pool/Math.sol#301)
FixedPoint.exp(int256) (contracts/mosaic-alpha-contracts/Pool/Math.sol#241-376) performs a multiplication on the result of a division:
    -product = (product * a5) / ONE_20 (contracts/mosaic-alpha-contracts/Pool/Math.sol#301)
    -product = (product * a6) / ONE_20 (contracts/mosaic-alpha-contracts/Pool/Math.sol#305)
FixedPoint.exp(int256) (contracts/mosaic-alpha-contracts/Pool/Math.sol#241-376) performs a multiplication on the result of a division:
    -product = (product * a6) / ONE_20 (contracts/mosaic-alpha-contracts/Pool/Math.sol#305)
    -product = (product * a7) / ONE_20 (contracts/mosaic-alpha-contracts/Pool/Math.sol#309)
FixedPoint.exp(int256) (contracts/mosaic-alpha-contracts/Pool/Math.sol#241-376) performs a multiplication on the result of a division:
    -product = (product * a7) / ONE_20 (contracts/mosaic-alpha-contracts/Pool/Math.sol#309)
    -product = (product * a8) / ONE_20 (contracts/mosaic-alpha-contracts/Pool/Math.sol#313)
```

```
FixedPoint.exp(int256) (contracts/mosaic-alpha-contracts/Pool/Math.sol#241-376) performs a multiplication on the result of a division:
    -product = (product * a8) / ONE_20 (contracts/mosaic-alpha-contracts/Pool/Math.sol#313)
    -product = (product * a9) / ONE_20 (contracts/mosaic-alpha-contracts/Pool/Math.sol#317)
FixedPoint.exp(int256) (contracts/mosaic-alpha-contracts/Pool/Math.sol#241-376) performs a multiplication on the result of a division:
    -term = ((term * x) / ONE_20) / 2 (contracts/mosaic-alpha-contracts/Pool/Math.sol#335)
    -term = ((term * x) / ONE_20) / 3 (contracts/mosaic-alpha-contracts/Pool/Math.sol#338)
FixedPoint.exp(int256) (contracts/mosaic-alpha-contracts/Pool/Math.sol#241-376) performs a multiplication on the result of a division:
    -term = ((term * x) / ONE_20) / 3 (contracts/mosaic-alpha-contracts/Pool/Math.sol#338)
    -term = ((term * x) / ONE_20) / 4 (contracts/mosaic-alpha-contracts/Pool/Math.sol#341)
FixedPoint.exp(int256) (contracts/mosaic-alpha-contracts/Pool/Math.sol#241-376) performs a multiplication on the result of a division:
    -term = ((term * x) / ONE_20) / 4 (contracts/mosaic-alpha-contracts/Pool/Math.sol#341)
    -term = ((term * x) / ONE_20) / 5 (contracts/mosaic-alpha-contracts/Pool/Math.sol#344)
FixedPoint.exp(int256) (contracts/mosaic-alpha-contracts/Pool/Math.sol#241-376) performs a multiplication on the result of a division:
    -term = ((term * x) / ONE_20) / 5 (contracts/mosaic-alpha-contracts/Pool/Math.sol#344)
    -term = ((term * x) / ONE_20) / 6 (contracts/mosaic-alpha-contracts/Pool/Math.sol#347)
FixedPoint.exp(int256) (contracts/mosaic-alpha-contracts/Pool/Math.sol#241-376) performs a multiplication on the result of a division:
    -term = ((term * x) / ONE_20) / 6 (contracts/mosaic-alpha-contracts/Pool/Math.sol#347)
    -term = ((term * x) / ONE_20) / 7 (contracts/mosaic-alpha-contracts/Pool/Math.sol#350)
FixedPoint.exp(int256) (contracts/mosaic-alpha-contracts/Pool/Math.sol#241-376) performs a multiplication on the result of a division:
    -term = ((term * x) / ONE_20) / 7 (contracts/mosaic-alpha-contracts/Pool/Math.sol#350)
    -term = ((term * x) / ONE_20) / 8 (contracts/mosaic-alpha-contracts/Pool/Math.sol#353)
FixedPoint.exp(int256) (contracts/mosaic-alpha-contracts/Pool/Math.sol#241-376) performs a multiplication on the result of a division:
    -term = ((term * x) / ONE_20) / 8 (contracts/mosaic-alpha-contracts/Pool/Math.sol#353)
    -term = ((term * x) / ONE_20) / 9 (contracts/mosaic-alpha-contracts/Pool/Math.sol#356)
FixedPoint.exp(int256) (contracts/mosaic-alpha-contracts/Pool/Math.sol#241-376) performs a multiplication on the result of a division:
    -term = ((term * x) / ONE_20) / 9 (contracts/mosaic-alpha-contracts/Pool/Math.sol#356)
    -term = ((term * x) / ONE_20) / 10 (contracts/mosaic-alpha-contracts/Pool/Math.sol#359)
FixedPoint.exp(int256) (contracts/mosaic-alpha-contracts/Pool/Math.sol#241-376) performs a multiplication on the result of a division:
    -term = ((term * x) / ONE_20) / 10 (contracts/mosaic-alpha-contracts/Pool/Math.sol#359)
    -term = ((term * x) / ONE_20) / 11 (contracts/mosaic-alpha-contracts/Pool/Math.sol#362)
FixedPoint.exp(int256) (contracts/mosaic-alpha-contracts/Pool/Math.sol#241-376) performs a multiplication on the result of a division:
    -term = ((term * x) / ONE_20) / 11 (contracts/mosaic-alpha-contracts/Pool/Math.sol#362)
    -term = ((term * x) / ONE_20) / 12 (contracts/mosaic-alpha-contracts/Pool/Math.sol#365)
FixedPoint.exp(int256) (contracts/mosaic-alpha-contracts/Pool/Math.sol#241-376) performs a multiplication on the result of a division:
    -(((product * seriesSum) / ONE_20) * firstAN) / 100 (contracts/mosaic-alpha-contracts/Pool/Math.sol#375)
FixedPoint._ln(int256) (contracts/mosaic-alpha-contracts/Pool/Math.sol#421-553) performs a multiplication on the result of a division:
    -a /= a1 (contracts/mosaic-alpha-contracts/Pool/Math.sol#451)
    -a *= 100 (contracts/mosaic-alpha-contracts/Pool/Math.sol#457)
FixedPoint._ln(int256) (contracts/mosaic-alpha-contracts/Pool/Math.sol#421-553) performs a multiplication on the result of a division:
    -a /= a1 (contracts/mosaic-alpha-contracts/Pool/Math.sol#451)
    -a = (a * ONE_20) / a2 (contracts/mosaic-alpha-contracts/Pool/Math.sol#462)
FixedPoint._ln(int256) (contracts/mosaic-alpha-contracts/Pool/Math.sol#421-553) performs a multiplication on the result of a division:
    -a = (a * ONE_20) / a2 (contracts/mosaic-alpha-contracts/Pool/Math.sol#462)
    -a = (a * ONE_20) / a3 (contracts/mosaic-alpha-contracts/Pool/Math.sol#467)
FixedPoint._ln(int256) (contracts/mosaic-alpha-contracts/Pool/Math.sol#421-553) performs a multiplication on the result of a division:
    -a = (a * ONE_20) / a3 (contracts/mosaic-alpha-contracts/Pool/Math.sol#467)
    -a = (a * ONE_20) / a4 (contracts/mosaic-alpha-contracts/Pool/Math.sol#472)
FixedPoint._ln(int256) (contracts/mosaic-alpha-contracts/Pool/Math.sol#421-553) performs a multiplication on the result of a division:
    -a = (a * ONE_20) / a4 (contracts/mosaic-alpha-contracts/Pool/Math.sol#472)
    -a = (a * ONE_20) / a5 (contracts/mosaic-alpha-contracts/Pool/Math.sol#477)
FixedPoint._ln(int256) (contracts/mosaic-alpha-contracts/Pool/Math.sol#421-553) performs a multiplication on the result of a division:
    -a = (a * ONE_20) / a5 (contracts/mosaic-alpha-contracts/Pool/Math.sol#477)
    -a = (a * ONE_20) / a6 (contracts/mosaic-alpha-contracts/Pool/Math.sol#482)
FixedPoint._ln(int256) (contracts/mosaic-alpha-contracts/Pool/Math.sol#421-553) performs a multiplication on the result of a division:
    -a = (a * ONE_20) / a6 (contracts/mosaic-alpha-contracts/Pool/Math.sol#482)
    -a = (a * ONE_20) / a7 (contracts/mosaic-alpha-contracts/Pool/Math.sol#487)
FixedPoint._ln(int256) (contracts/mosaic-alpha-contracts/Pool/Math.sol#421-553) performs a multiplication on the result of a division:
    -a = (a * ONE_20) / a7 (contracts/mosaic-alpha-contracts/Pool/Math.sol#487)
    -a = (a * ONE_20) / a8 (contracts/mosaic-alpha-contracts/Pool/Math.sol#492)
FixedPoint._ln(int256) (contracts/mosaic-alpha-contracts/Pool/Math.sol#421-553) performs a multiplication on the result of a division:
    -a = (a * ONE_20) / a8 (contracts/mosaic-alpha-contracts/Pool/Math.sol#492)
    -a = (a * ONE_20) / a9 (contracts/mosaic-alpha-contracts/Pool/Math.sol#497)
FixedPoint._ln(int256) (contracts/mosaic-alpha-contracts/Pool/Math.sol#421-553) performs a multiplication on the result of a division:
    -a = (a * ONE_20) / a9 (contracts/mosaic-alpha-contracts/Pool/Math.sol#497)
    -a = (a * ONE_20) / a10 (contracts/mosaic-alpha-contracts/Pool/Math.sol#502)
FixedPoint._ln(int256) (contracts/mosaic-alpha-contracts/Pool/Math.sol#421-553) performs a multiplication on the result of a division:
    -a = (a * ONE_20) / a10 (contracts/mosaic-alpha-contracts/Pool/Math.sol#502)
    -a = (a * ONE_20) / a11 (contracts/mosaic-alpha-contracts/Pool/Math.sol#507)
FixedPoint._ln(int256) (contracts/mosaic-alpha-contracts/Pool/Math.sol#421-553) performs a multiplication on the result of a division:
    -z = ((a - ONE_20) * ONE_20) / (a + ONE_20) (contracts/mosaic-alpha-contracts/Pool/Math.sol#518)
    -z_squared = (z * z) / ONE_20 (contracts/mosaic-alpha-contracts/Pool/Math.sol#519)
FixedPoint._ln(int256) (contracts/mosaic-alpha-contracts/Pool/Math.sol#421-553) performs a multiplication on the result of a division:
    -z = ((a - ONE_20) * ONE_20) / (a + ONE_20) (contracts/mosaic-alpha-contracts/Pool/Math.sol#518)
    -z_squared = (z * z) / ONE_20 (contracts/mosaic-alpha-contracts/Pool/Math.sol#519)
    -num = z (contracts/mosaic-alpha-contracts/Pool/Math.sol#522)
    -num = (num * z_squared) / ONE_20 (contracts/mosaic-alpha-contracts/Pool/Math.sol#528)
FixedPoint._ln(int256) (contracts/mosaic-alpha-contracts/Pool/Math.sol#421-553) performs a multiplication on the result of a division:
    -z_squared = (z * z) / ONE_20 (contracts/mosaic-alpha-contracts/Pool/Math.sol#519)
    -num = (num * z_squared) / ONE_20 (contracts/mosaic-alpha-contracts/Pool/Math.sol#528)
    -num = (num * z_squared) / ONE_20 (contracts/mosaic-alpha-contracts/Pool/Math.sol#531)
FixedPoint._ln(int256) (contracts/mosaic-alpha-contracts/Pool/Math.sol#421-553) performs a multiplication on the result of a division:
    -z_squared = (z * z) / ONE_20 (contracts/mosaic-alpha-contracts/Pool/Math.sol#519)
    -num = (num * z_squared) / ONE_20 (contracts/mosaic-alpha-contracts/Pool/Math.sol#531)
    -num = (num * z_squared) / ONE_20 (contracts/mosaic-alpha-contracts/Pool/Math.sol#534)
FixedPoint._ln(int256) (contracts/mosaic-alpha-contracts/Pool/Math.sol#421-553) performs a multiplication on the result of a division:
    -z_squared = (z * z) / ONE_20 (contracts/mosaic-alpha-contracts/Pool/Math.sol#519)
    -num = (num * z_squared) / ONE_20 (contracts/mosaic-alpha-contracts/Pool/Math.sol#534)
    -num = (num * z_squared) / ONE_20 (contracts/mosaic-alpha-contracts/Pool/Math.sol#537)
FixedPoint._ln(int256) (contracts/mosaic-alpha-contracts/Pool/Math.sol#421-553) performs a multiplication on the result of a division:
    -z_squared = (z * z) / ONE_20 (contracts/mosaic-alpha-contracts/Pool/Math.sol#519)
    -num = (num * z_squared) / ONE_20 (contracts/mosaic-alpha-contracts/Pool/Math.sol#537)
    -num = (num * z_squared) / ONE_20 (contracts/mosaic-alpha-contracts/Pool/Math.sol#540)
FixedPoint._ln(int256) (contracts/mosaic-alpha-contracts/Pool/Math.sol#421-553) performs a multiplication on the result of a division:
    -z = ((a - ONE_20) * ONE_20) / (a + ONE_20) (contracts/mosaic-alpha-contracts/Pool/Math.sol#518)
    -num = z (contracts/mosaic-alpha-contracts/Pool/Math.sol#522)
    -seriesSum = num (contracts/mosaic-alpha-contracts/Pool/Math.sol#525)
    -seriesSum *= 2 (contracts/mosaic-alpha-contracts/Pool/Math.sol#546)
FixedPoint._ln_36(int256) (contracts/mosaic-alpha-contracts/Pool/Math.sol#561-608) performs a multiplication on the result of a division:
    -z = ((x - ONE_36) * ONE_36) / (x + ONE_36) (contracts/mosaic-alpha-contracts/Pool/Math.sol#573)
    -z_squared = (z * z) / ONE_36 (contracts/mosaic-alpha-contracts/Pool/Math.sol#574)
FixedPoint._ln_36(int256) (contracts/mosaic-alpha-contracts/Pool/Math.sol#561-608) performs a multiplication on the result of a division:
    -z = ((x - ONE_36) * ONE_36) / (x + ONE_36) (contracts/mosaic-alpha-contracts/Pool/Math.sol#573)
    -z_squared = (z * z) / ONE_36 (contracts/mosaic-alpha-contracts/Pool/Math.sol#574)
    -num = z (contracts/mosaic-alpha-contracts/Pool/Math.sol#577)
    -num = (num * z_squared) / ONE_36 (contracts/mosaic-alpha-contracts/Pool/Math.sol#583)
FixedPoint._ln_36(int256) (contracts/mosaic-alpha-contracts/Pool/Math.sol#561-608) performs a multiplication on the result of a division:
    -z_squared = (z * z) / ONE_36 (contracts/mosaic-alpha-contracts/Pool/Math.sol#574)
    -num = (num * z_squared) / ONE_36 (contracts/mosaic-alpha-contracts/Pool/Math.sol#583)
    -num = (num * z_squared) / ONE_36 (contracts/mosaic-alpha-contracts/Pool/Math.sol#586)
FixedPoint._ln_36(int256) (contracts/mosaic-alpha-contracts/Pool/Math.sol#561-608) performs a multiplication on the result of a division:
    -z_squared = (z * z) / ONE_36 (contracts/mosaic-alpha-contracts/Pool/Math.sol#574)
    -num = (num * z_squared) / ONE_36 (contracts/mosaic-alpha-contracts/Pool/Math.sol#586)
    -num = (num * z_squared) / ONE_36 (contracts/mosaic-alpha-contracts/Pool/Math.sol#589)
FixedPoint._ln_36(int256) (contracts/mosaic-alpha-contracts/Pool/Math.sol#561-608) performs a multiplication on the result of a division:
    -z_squared = (z * z) / ONE_36 (contracts/mosaic-alpha-contracts/Pool/Math.sol#574)
    -num = (num * z_squared) / ONE_36 (contracts/mosaic-alpha-contracts/Pool/Math.sol#589)
    -num = (num * z_squared) / ONE_36 (contracts/mosaic-alpha-contracts/Pool/Math.sol#592)
```

```
FixedPoint._ln_36(int256) (contracts/mosaic-alpha-contracts/Pool/Math.sol#561-608) performs a multiplication on the result of a division:
    -z_squared = (z * z) / ONE_36 (contracts/mosaic-alpha-contracts/Pool/Math.sol#574)
    -num = (num * z_squared) / ONE_36 (contracts/mosaic-alpha-contracts/Pool/Math.sol#592)
    -num = (num * z_squared) / ONE_36 (contracts/mosaic-alpha-contracts/Pool/Math.sol#595)
FixedPoint._ln_36(int256) (contracts/mosaic-alpha-contracts/Pool/Math.sol#561-608) performs a multiplication on the result of a division:
    -z_squared = (z * z) / ONE_36 (contracts/mosaic-alpha-contracts/Pool/Math.sol#574)
    -num = (num * z_squared) / ONE_36 (contracts/mosaic-alpha-contracts/Pool/Math.sol#595)
    -num = (num * z_squared) / ONE_36 (contracts/mosaic-alpha-contracts/Pool/Math.sol#598)
FixedPoint._ln_36(int256) (contracts/mosaic-alpha-contracts/Pool/Math.sol#561-608) performs a multiplication on the result of a division:
    -z_squared = (z * z) / ONE_36 (contracts/mosaic-alpha-contracts/Pool/Math.sol#574)
    -num = (num * z_squared) / ONE_36 (contracts/mosaic-alpha-contracts/Pool/Math.sol#598)
    -num = (num * z_squared) / ONE_36 (contracts/mosaic-alpha-contracts/Pool/Math.sol#601)
FixedPoint._ln_36(int256) (contracts/mosaic-alpha-contracts/Pool/Math.sol#561-608) performs a multiplication on the result of a division:
    -z = ((x - ONE_36) * ONE_36) / (x + ONE_36) (contracts/mosaic-alpha-contracts/Pool/Math.sol#573)
    -num = z (contracts/mosaic-alpha-contracts/Pool/Math.sol#577)
    -seriesSum = num (contracts/mosaic-alpha-contracts/Pool/Math.sol#580)
    -seriesSum * 2 (contracts/mosaic-alpha-contracts/Pool/Math.sol#607)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply

Vault.bid(uint256,uint256,bool,bool,uint256) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1163-1213) performs a multiplication on the result of a
division:
    -_toPay = _currentPrice * amount / _auctionDetails.remainingAmount (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1193)
    -_toPay += _toPay * vaultState.swapFee / 10000 (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1195)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply

FullMath.fullDiv(uint256,uint256,uint256) (contracts/mosaic-alpha-contracts/helpers/pancake/libraries/FullMath.sol#14-35) performs a multiplication on the
result of a division:
    -d /= pow2 (contracts/mosaic-alpha-contracts/helpers/pancake/libraries/FullMath.sol#21)
    -r *= 2 - d * r (contracts/mosaic-alpha-contracts/helpers/pancake/libraries/FullMath.sol#26)
FullMath.fullDiv(uint256,uint256,uint256) (contracts/mosaic-alpha-contracts/helpers/pancake/libraries/FullMath.sol#14-35) performs a multiplication on the
result of a division:
    -d /= pow2 (contracts/mosaic-alpha-contracts/helpers/pancake/libraries/FullMath.sol#21)
    -r *= 2 - d * r (contracts/mosaic-alpha-contracts/helpers/pancake/libraries/FullMath.sol#27)
FullMath.fullDiv(uint256,uint256,uint256) (contracts/mosaic-alpha-contracts/helpers/pancake/libraries/FullMath.sol#14-35) performs a multiplication on the
result of a division:
    -d /= pow2 (contracts/mosaic-alpha-contracts/helpers/pancake/libraries/FullMath.sol#21)
    -r *= 2 - d * r (contracts/mosaic-alpha-contracts/helpers/pancake/libraries/FullMath.sol#28)
FullMath.fullDiv(uint256,uint256,uint256) (contracts/mosaic-alpha-contracts/helpers/pancake/libraries/FullMath.sol#14-35) performs a multiplication on the
result of a division:
    -d /= pow2 (contracts/mosaic-alpha-contracts/helpers/pancake/libraries/FullMath.sol#21)
    -r *= 2 - d * r (contracts/mosaic-alpha-contracts/helpers/pancake/libraries/FullMath.sol#29)
FullMath.fullDiv(uint256,uint256,uint256) (contracts/mosaic-alpha-contracts/helpers/pancake/libraries/FullMath.sol#14-35) performs a multiplication on the
result of a division:
    -d /= pow2 (contracts/mosaic-alpha-contracts/helpers/pancake/libraries/FullMath.sol#21)
    -r *= 2 - d * r (contracts/mosaic-alpha-contracts/helpers/pancake/libraries/FullMath.sol#30)
FullMath.fullDiv(uint256,uint256,uint256) (contracts/mosaic-alpha-contracts/helpers/pancake/libraries/FullMath.sol#14-35) performs a multiplication on the
result of a division:
    -d /= pow2 (contracts/mosaic-alpha-contracts/helpers/pancake/libraries/FullMath.sol#21)
    -r *= 2 - d * r (contracts/mosaic-alpha-contracts/helpers/pancake/libraries/FullMath.sol#31)
FullMath.fullDiv(uint256,uint256,uint256) (contracts/mosaic-alpha-contracts/helpers/pancake/libraries/FullMath.sol#14-35) performs a multiplication on the
result of a division:
    -d /= pow2 (contracts/mosaic-alpha-contracts/helpers/pancake/libraries/FullMath.sol#21)
    -r *= 2 - d * r (contracts/mosaic-alpha-contracts/helpers/pancake/libraries/FullMath.sol#32)
FullMath.fullDiv(uint256,uint256,uint256) (contracts/mosaic-alpha-contracts/helpers/pancake/libraries/FullMath.sol#14-35) performs a multiplication on the
result of a division:
    -d /= pow2 (contracts/mosaic-alpha-contracts/helpers/pancake/libraries/FullMath.sol#21)
    -r *= 2 - d * r (contracts/mosaic-alpha-contracts/helpers/pancake/libraries/FullMath.sol#33)
FullMath.fullDiv(uint256,uint256,uint256) (contracts/mosaic-alpha-contracts/helpers/pancake/libraries/FullMath.sol#14-35) performs a multiplication on the
result of a division:
    -l /= pow2 (contracts/mosaic-alpha-contracts/helpers/pancake/libraries/FullMath.sol#22)
    -l * r (contracts/mosaic-alpha-contracts/helpers/pancake/libraries/FullMath.sol#34)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply
```

incorrect-erc721-interface

```
Pool (contracts/mosaic-alpha-contracts/Pool/Pool.sol#15-845) has incorrect ERC721 function interface:IPool.transferFrom(address,address,uint256)
(contracts/mosaic-alpha-contracts/Interfaces/IPool.sol#39)
Pool (contracts/mosaic-alpha-contracts/Pool/Pool.sol#15-845) has incorrect ERC721 function interface:IPool.safeTransferFrom(address,address,uint256)
(contracts/mosaic-alpha-contracts/Interfaces/IPool.sol#40)
Pool (contracts/mosaic-alpha-contracts/Pool/Pool.sol#15-845) has incorrect ERC721 function interface:IPool.approve(address,uint256) (contracts/mosaic-
alpha-contracts/Interfaces/IPool.sol#41)
Pool (contracts/mosaic-alpha-contracts/Pool/Pool.sol#15-845) has incorrect ERC721 function interface:Pool.transferFrom(address,address,uint256)
(contracts/mosaic-alpha-contracts/Pool/Pool.sol#295-297)
Pool (contracts/mosaic-alpha-contracts/Pool/Pool.sol#15-845) has incorrect ERC721 function interface:Pool.safeTransferFrom(address,address,uint256)
(contracts/mosaic-alpha-contracts/Pool/Pool.sol#299-301)
Pool (contracts/mosaic-alpha-contracts/Pool/Pool.sol#15-845) has incorrect ERC721 function interface:Pool.approve(address,uint256) (contracts/mosaic-alpha-
contracts/Pool/Pool.sol#319-324)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-erc721-interface

IPool (contracts/mosaic-alpha-contracts/Interfaces/IPool.sol#7-132) has incorrect ERC721 function interface:IPool.transferFrom(address,address,uint256)
(contracts/mosaic-alpha-contracts/Interfaces/IPool.sol#39)
IPool (contracts/mosaic-alpha-contracts/Interfaces/IPool.sol#7-132) has incorrect ERC721 function interface:IPool.safeTransferFrom(address,address,uint256)
(contracts/mosaic-alpha-contracts/Interfaces/IPool.sol#40)
IPool (contracts/mosaic-alpha-contracts/Interfaces/IPool.sol#7-132) has incorrect ERC721 function interface:IPool.approve(address,uint256)
(contracts/mosaic-alpha-contracts/Interfaces/IPool.sol#41)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-erc721-interface
```

dangerous-strict-equalities

```
PancakePair._safeTransfer(address,address,uint256) (contracts/mosaic-alpha-contracts/helpers/pancake/PancakePair.sol#46-49) uses a dangerous strict
equality:
    - require(bool,string)(success && (data.length == 0 || abi.decode(data,(bool))),Pancake: TRANSFER_FAILED) (contracts/mosaic-alpha-
contracts/helpers/pancake/PancakePair.sol#48)
PancakePair.mint(address) (contracts/mosaic-alpha-contracts/helpers/pancake/PancakePair.sol#112-133) uses a dangerous strict equality:
    - _totalSupply == 0 (contracts/mosaic-alpha-contracts/helpers/pancake/PancakePair.sol#121)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities

Governance.propose_action(uint256,address,bytes) (contracts/mosaic-alpha-contracts/Governance/Governance.sol#417-430) uses a dangerous strict equality:
    - require(bool,string)(action_time_limit[_id] == 0,Create a new one) (contracts/mosaic-alpha-contracts/Governance/Governance.sol#420)
Governance.propose_config(string,bool,address,address[],bool[],address[],uint256[],bytes32[]) (contracts/mosaic-alpha-
contracts/Governance/Governance.sol#229-260) uses a dangerous strict equality:
    - require(bool,string)(conf_time_limit[conf_counter] == 0,In progress) (contracts/mosaic-alpha-contracts/Governance/Governance.sol#242)
Governance.propose_core_change(address,bool,address[],address[]) (contracts/mosaic-alpha-contracts/Governance/Governance.sol#263-279) uses a dangerous
strict equality:
    - require(bool,string)(conf_time_limit[conf_counter] == 0,In progress) (contracts/mosaic-alpha-contracts/Governance/Governance.sol#266)
Vault._getBidPrice(uint256,uint32) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1130-1151) uses a dangerous strict equality:
    - _now >= _end || remainingAmount == 0 (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1141)
Vault.isRunning(uint256) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1059-1065) uses a dangerous strict equality:
    - auctions[auctionId].remainingAmount == 0 || _expiryTime < block.timestamp (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1061)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities

Fees._updateUserTimeIntegrals(address,uint32,uint8,uint32) (contracts/mosaic-alpha-contracts/Fees/Fees.sol#185-197) uses a dangerous strict equality:
    - timeDelta == 0 (contracts/mosaic-alpha-contracts/Fees/Fees.sol#187)
Fees.claimUserClaimableFee(address,address) (contracts/mosaic-alpha-contracts/Fees/Fees.sol#205-305) uses a dangerous strict equality:
```

```
    - timeDelta == 0 (contracts/mosaic-alpha-contracts/Fees/Fees.sol#207)
Fees.claimUserClaimableFee(address,address) (contracts/mosaic-alpha-contracts/Fees/Fees.sol#205-305) uses a dangerous strict equality:
    - timeDelta == block.timestamp (contracts/mosaic-alpha-contracts/Fees/Fees.sol#245)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities

AffiliateBooster.pickNextReferral() (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#372-410) uses a dangerous strict equality:
    - state.totalPlayerWeight == 0 (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#374)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities
```

reentrancy-vulnerabilities-1

```
Reentrancy in PancakePair.burn(address) (contracts/mosaic-alpha-contracts/helpers/pancake/PancakePair.sol#136-158):
    External calls:
    - _safeTransfer(_token0,to,amount0) (contracts/mosaic-alpha-contracts/helpers/pancake/PancakePair.sol#150)
        - (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value)) (contracts/mosaic-alpha-contracts/helpers/pancake/PancakePair.sol#47)
    - _safeTransfer(_token1,to,amount1) (contracts/mosaic-alpha-contracts/helpers/pancake/PancakePair.sol#151)
        - (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value)) (contracts/mosaic-alpha-contracts/helpers/pancake/PancakePair.sol#47)
    State variables written after the call(s):
    - _update(balance0,balance1,_reserve0,_reserve1) (contracts/mosaic-alpha-contracts/helpers/pancake/PancakePair.sol#155)
        - blockTimestampLast = blockTimestamp (contracts/mosaic-alpha-contracts/helpers/pancake/PancakePair.sol#86)
    - kLast = uint256(reserve0).mul(reserve1) (contracts/mosaic-alpha-contracts/helpers/pancake/PancakePair.sol#156)
    - _update(balance0,balance1,_reserve0,_reserve1) (contracts/mosaic-alpha-contracts/helpers/pancake/PancakePair.sol#155)
        - reserve0 = uint112(balance0) (contracts/mosaic-alpha-contracts/helpers/pancake/PancakePair.sol#84)
    - _update(balance0,balance1,_reserve0,_reserve1) (contracts/mosaic-alpha-contracts/helpers/pancake/PancakePair.sol#155)
        - reserve1 = uint112(balance1) (contracts/mosaic-alpha-contracts/helpers/pancake/PancakePair.sol#85)
Reentrancy in PancakeFactory.createPair(address,address) (contracts/mosaic-alpha-contracts/helpers/pancake/PancakeFactory.sol#29-44):
    External calls:
    - IPancakePair(pair).initialize(token0,token1) (contracts/mosaic-alpha-contracts/helpers/pancake/PancakeFactory.sol#39)
    State variables written after the call(s):
    - getPair[token0][token1] = pair (contracts/mosaic-alpha-contracts/helpers/pancake/PancakeFactory.sol#40)
    - getPair[token1][token0] = pair (contracts/mosaic-alpha-contracts/helpers/pancake/PancakeFactory.sol#41)
Reentrancy in PancakePair.swap(uint256,uint256,address,bytes) (contracts/mosaic-alpha-contracts/helpers/pancake/PancakePair.sol#161-189):
    External calls:
    - _safeTransfer(_token0,to,amount0Out) (contracts/mosaic-alpha-contracts/helpers/pancake/PancakePair.sol#172)
        - (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value)) (contracts/mosaic-alpha-contracts/helpers/pancake/PancakePair.sol#47)
    - _safeTransfer(_token1,to,amount1Out) (contracts/mosaic-alpha-contracts/helpers/pancake/PancakePair.sol#173)
        - (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value)) (contracts/mosaic-alpha-contracts/helpers/pancake/PancakePair.sol#47)
    - IPancakeCallee(to).pancakeCall(msg.sender,amount0Out,amount1Out,data) (contracts/mosaic-alpha-contracts/helpers/pancake/PancakePair.sol#174)
    State variables written after the call(s):
    - _update(balance0,balance1,_reserve0,_reserve1) (contracts/mosaic-alpha-contracts/helpers/pancake/PancakePair.sol#187)
        - blockTimestampLast = blockTimestamp (contracts/mosaic-alpha-contracts/helpers/pancake/PancakePair.sol#86)
    - _update(balance0,balance1,_reserve0,_reserve1) (contracts/mosaic-alpha-contracts/helpers/pancake/PancakePair.sol#187)
        - reserve0 = uint112(balance0) (contracts/mosaic-alpha-contracts/helpers/pancake/PancakePair.sol#84)
    - _update(balance0,balance1,_reserve0,_reserve1) (contracts/mosaic-alpha-contracts/helpers/pancake/PancakePair.sol#187)
        - reserve1 = uint112(balance1) (contracts/mosaic-alpha-contracts/helpers/pancake/PancakePair.sol#85)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1

Reentrancy in Pool._updatePerfFeePricePerLp(IVault.TotalSupplyBase,uint256) (contracts/mosaic-alpha-contracts/Pool/Pool.sol#814-822):
    External calls:
    - feeAmount = (ts.amount * (valuePerLp - lastValuePerLp) * uint256(poolFees.performanceFee) / lastValuePerLp / 10000).toUint128() (contracts/mosaic-
alpha-contracts/Pool/Pool.sol#819)
    State variables written after the call(s):
    - lastValuePerLp = valuePerLp.toUint128() (contracts/mosaic-alpha-contracts/Pool/Pool.sol#820)
Reentrancy in Pool.dutchAuction(address,uint256,address,uint32,uint32,uint256) (contracts/mosaic-alpha-contracts/Pool/Pool.sol#627-655):
    External calls:
    - _auctionId = IVault(vaultAddress).startAuction(tokenToSell,amountToSell,tokenToBuy,duration,expiration,endingPrice) (contracts/mosaic-alpha-
contracts/Pool/Pool.sol#648)
    State variables written after the call(s):
    - runningDutchAuctionIds.push(_auctionId) (contracts/mosaic-alpha-contracts/Pool/Pool.sol#651)
Reentrancy in Pool.removeToken(uint256) (contracts/mosaic-alpha-contracts/Pool/Pool.sol#569-590):
    External calls:
    - IVault(vaultAddress).deregisterToken(tokens[_index],_remainingBalance) (contracts/mosaic-alpha-contracts/Pool/Pool.sol#581)
    State variables written after the call(s):
    - tokens[_index] = tokens[tokens.length - 1] (contracts/mosaic-alpha-contracts/Pool/Pool.sol#583)
    - tokens.pop() (contracts/mosaic-alpha-contracts/Pool/Pool.sol#584)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1

Reentrancy in Vault.Burn(uint32,uint256,uint256[],bool,uint256,address) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#913-951):
    External calls:
    - (amountsToReceive,trailingFeeLpTokens) = IPool(_poolAddr)._burn(LPTokensToBurn,poolStates[_poolAddr].totalSupplyBase) (contracts/mosaic-alpha-
contracts/Vault/Vault.sol#921)
    - _transferInternalBalance(tokens[i],_poolAddr,msg.sender,amountsToReceive[i]) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#927)
        - IMosaicOracle(mosaicOracle).consultUsdPoolPricePerLp(poolAddr,ts) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#703)
        - IFees(fees).onBeforeTransfer(_pool,_from,_to,_internalBalance[_from][_pool],_internalBalance[_to]
[_pool],amount,originalTotalsupplyBase,trader,payload) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#598-601)
        - IPool(token).emitTransfer(from,to,amount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#499)
        - performanceFeesToMint = (IPool(_pool).updatePerfFeePricePerLp(tsTmp,valuePerLp)) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#568)
        - IPool(_pool).emitTransfer(address(0),fees,amount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#610)
        - IFees(fees).onBeforeTransfer(_pool,_from,_to,_internalBalance[_from][_pool],_internalBalance[_to][_pool],amount,0,address(0),payload)
(contracts/mosaic-alpha-contracts/Vault/Vault.sol#614)
    State variables written after the call(s):
    - _transferInternalBalance(tokens[i],_poolAddr,msg.sender,amountsToReceive[i]) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#927)
        - poolStates[_pool].lastTrailingTimestamp = uint48(block.timestamp) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#560)
        - poolStates[_pool].totalSupplyBase.amount += uint224(expAmount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#603)
        - poolStates[_pool].totalSupplyBase.timestamp = (block.timestamp).toUint32() (contracts/mosaic-alpha-contracts/Vault/Vault.sol#604)
        - poolStates[_pool].lastPerformanceTimestamp = uint48(block.timestamp) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#570)
Reentrancy in Vault.Burn(uint32,uint256,uint256[],bool,uint256,address) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#913-951):
    External calls:
    - (amountsToReceive,trailingFeeLpTokens) = IPool(_poolAddr)._burn(LPTokensToBurn,poolStates[_poolAddr].totalSupplyBase) (contracts/mosaic-alpha-
contracts/Vault/Vault.sol#921)
    - _burnFromInternalBalance(_poolAddr,msg.sender,LPTokensToBurn,true,true) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#938)
        - IMosaicOracle(mosaicOracle).consultUsdPoolPricePerLp(poolAddr,ts) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#703)
        - IPool(token).emitTransfer(from,address(0),amount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#536)
        - IFees(fees).onBeforeTransfer(_pool,_from,_to,_internalBalance[_from][_pool],_internalBalance[_to]
[_pool],amount,originalTotalsupplyBase,trader,payload) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#598-601)
        - performanceFeesToMint = (IPool(_pool).updatePerfFeePricePerLp(tsTmp,valuePerLp)) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#568)
        - IPool(_pool).emitTransfer(address(0),fees,amount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#610)
        - IFees(fees).onBeforeTransfer(_pool,_from,_to,_internalBalance[_from][_pool],_internalBalance[_to][_pool],amount,0,address(0),payload)
(contracts/mosaic-alpha-contracts/Vault/Vault.sol#614)
    State variables written after the call(s):
    - _burnFromInternalBalance(_poolAddr,msg.sender,LPTokensToBurn,true,true) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#938)
        - poolStates[token].totalSupplyBase.amount -= (amount).toUint224() (contracts/mosaic-alpha-contracts/Vault/Vault.sol#531)
        - poolStates[token].totalSupplyBase.timestamp = uint32(block.timestamp) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#532)
        - poolStates[_pool].lastTrailingTimestamp = uint48(block.timestamp) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#560)
        - poolStates[_pool].totalSupplyBase.amount += uint224(expAmount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#603)
        - poolStates[_pool].totalSupplyBase.timestamp = (block.timestamp).toUint32() (contracts/mosaic-alpha-contracts/Vault/Vault.sol#604)
        - poolStates[_pool].lastPerformanceTimestamp = uint48(block.timestamp) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#570)
Reentrancy in Vault.Burn(uint32,uint256,uint256[],bool,uint256,address) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#913-951):
    External calls:
    - (amountsToReceive,trailingFeeLpTokens) = IPool(_poolAddr)._burn(LPTokensToBurn,poolStates[_poolAddr].totalSupplyBase) (contracts/mosaic-alpha-
contracts/Vault/Vault.sol#921)
    - (trailing,performance) = _calculateTrailingAndPerformanceFee(_poolAddr,true,false) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#941)
        - IMosaicOracle(mosaicOracle).consultUsdPoolPricePerLp(poolAddr,ts) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#703)
        - performanceFeesToMint = (IPool(_pool).updatePerfFeePricePerLp(tsTmp,valuePerLp)) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#568)
    State variables written after the call(s):
    - (trailing,performance) = _calculateTrailingAndPerformanceFee(_poolAddr,true,false) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#941)
```

```
        - poolStates[_pool].lastTrailingTimestamp = uint48(block.timestamp) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#560)
        - poolStates[_pool].lastPerformanceTimestamp = uint48(block.timestamp) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#570)
Reentrancy in Vault.Burn(uint32,uint256,uint256[],bool,uint256,address) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#913-951):
    External calls:
    - (amountsToReceive,trailingFeeLpTokens) = IPool(_poolAddr)._burn(LPTokensToBurn,poolStates[_poolAddr].totalSupplyBase) (contracts/mosaic-alpha-
contracts/Vault/Vault.sol#921)
    - (trailing,performance) = _calculateTrailingAndPerformanceFee(_poolAddr,true,false) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#941)
        - IMosaicOracle(mosaicOracle).consultUsdPoolPricePerLp(poolAddr,ts) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#703)
        - performanceFeesToMint = (IPool(_pool).updatePerfFeePricePerLp(tsTmp,valuePerLp)) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#568)
    - IFees(fees).allocateTrailingFee(_poolAddr,trader,trailing,poolStates[_poolAddr].totalSupplyBase.amount,tx.origin) (contracts/mosaic-alpha-
contracts/Vault/Vault.sol#942)
    - IFees(fees).allocatePerformanceFee(_poolAddr,trader,performance,poolStates[_poolAddr].totalSupplyBase.amount,tx.origin) (contracts/mosaic-alpha-
contracts/Vault/Vault.sol#943)
    - _mintToInternalBalance(_poolAddr,fees,trailing + performance,true,true) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#944)
        - IMosaicOracle(mosaicOracle).consultUsdPoolPricePerLp(poolAddr,ts) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#703)
        - IPool(token).emitTransfer(address(0),to,amount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#521)
        - IFees(fees).onBeforeTransfer(_pool,_from,_to,_internalBalance[_from][_pool],_internalBalance[_to]
[_pool],amount,originalTotalsupplyBase,trader,payload) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#598-601)
        - performanceFeesToMint = (IPool(_pool).updatePerfFeePricePerLp(tsTmp,valuePerLp)) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#568)
        - IPool(_pool).emitTransfer(address(0),fees,amount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#610)
        - IFees(fees).onBeforeTransfer(_pool,_from,_to,_internalBalance[_from][_pool],_internalBalance[_to][_pool],amount,0,address(0),payload)
(contracts/mosaic-alpha-contracts/Vault/Vault.sol#614)
    State variables written after the call(s):
    - _mintToInternalBalance(_poolAddr,fees,trailing + performance,true,true) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#944)
        - poolStates[token].totalSupplyBase.amount += (amount).toUint224() (contracts/mosaic-alpha-contracts/Vault/Vault.sol#514)
        - poolStates[token].totalSupplyBase.timestamp = uint32(block.timestamp) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#515)
        - poolStates[_pool].lastTrailingTimestamp = uint48(block.timestamp) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#560)
        - poolStates[_pool].totalSupplyBase.amount += uint224(expAmount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#603)
        - poolStates[_pool].totalSupplyBase.timestamp = (block.timestamp).toUint32() (contracts/mosaic-alpha-contracts/Vault/Vault.sol#604)
        - poolStates[_pool].lastPerformanceTimestamp = uint48(block.timestamp) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#570)
Reentrancy in Vault.Burn(uint32,uint256,uint256[],bool,uint256,address) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#913-951):
    External calls:
    - (amountsToReceive,trailingFeeLpTokens) = IPool(_poolAddr)._burn(LPTokensToBurn,poolStates[_poolAddr].totalSupplyBase) (contracts/mosaic-alpha-
contracts/Vault/Vault.sol#921)
    - (trailing,performance) = _calculateTrailingAndPerformanceFee(_poolAddr,true,false) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#941)
        - IMosaicOracle(mosaicOracle).consultUsdPoolPricePerLp(poolAddr,ts) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#703)
        - performanceFeesToMint = (IPool(_pool).updatePerfFeePricePerLp(tsTmp,valuePerLp)) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#568)
    - IFees(fees).allocateTrailingFee(_poolAddr,trader,trailing,poolStates[_poolAddr].totalSupplyBase.amount,tx.origin) (contracts/mosaic-alpha-
contracts/Vault/Vault.sol#942)
    - IFees(fees).allocatePerformanceFee(_poolAddr,trader,performance,poolStates[_poolAddr].totalSupplyBase.amount,tx.origin) (contracts/mosaic-alpha-
contracts/Vault/Vault.sol#943)
    - _mintToInternalBalance(_poolAddr,fees,trailing + performance,true,true) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#944)
        - IMosaicOracle(mosaicOracle).consultUsdPoolPricePerLp(poolAddr,ts) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#703)
        - IPool(token).emitTransfer(address(0),to,amount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#521)
        - IFees(fees).onBeforeTransfer(_pool,_from,_to,_internalBalance[_from][_pool],_internalBalance[_to]
[_pool],amount,originalTotalsupplyBase,trader,payload) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#598-601)
        - performanceFeesToMint = (IPool(_pool).updatePerfFeePricePerLp(tsTmp,valuePerLp)) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#568)
        - IPool(_pool).emitTransfer(address(0),fees,amount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#610)
        - IFees(fees).onBeforeTransfer(_pool,_from,_to,_internalBalance[_from][_pool],_internalBalance[_to][_pool],amount,0,address(0),payload)
(contracts/mosaic-alpha-contracts/Vault/Vault.sol#614)
    - _burnFromInternalBalance(_poolAddr,msg.sender,LPTokensToBurn,true,true) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#946)
        - IMosaicOracle(mosaicOracle).consultUsdPoolPricePerLp(poolAddr,ts) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#703)
        - IPool(token).emitTransfer(from,address(0),amount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#536)
        - IFees(fees).onBeforeTransfer(_pool,_from,_to,_internalBalance[_from][_pool],_internalBalance[_to]
[_pool],amount,originalTotalsupplyBase,trader,payload) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#598-601)
        - performanceFeesToMint = (IPool(_pool).updatePerfFeePricePerLp(tsTmp,valuePerLp)) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#568)
        - IPool(_pool).emitTransfer(address(0),fees,amount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#610)
        - IFees(fees).onBeforeTransfer(_pool,_from,_to,_internalBalance[_from][_pool],_internalBalance[_to][_pool],amount,0,address(0),payload)
(contracts/mosaic-alpha-contracts/Vault/Vault.sol#614)
    State variables written after the call(s):
    - _burnFromInternalBalance(_poolAddr,msg.sender,LPTokensToBurn,true,true) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#946)
        - _internalBalance[from][token] -= amount (contracts/mosaic-alpha-contracts/Vault/Vault.sol#530)
        - _internalBalance[fees][_pool] += expAmount (contracts/mosaic-alpha-contracts/Vault/Vault.sol#606)
    - _burnFromInternalBalance(_poolAddr,msg.sender,LPTokensToBurn,true,true) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#946)
        - poolOfUser[user].push(poolId) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1226)
        - poolOfUser[user][index_scope_0] = replacementPoolId (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1240)
        - poolOfUser[user].pop() (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1241)
    - _burnFromInternalBalance(_poolAddr,msg.sender,LPTokensToBurn,true,true) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#946)
        - poolOfUserIndexes[user][poolId] = (poolOfUser[user].length).toUint32() (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1228)
        - poolOfUserIndexes[user][replacementPoolId] = uint32(index_scope_0) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1239)
        - poolOfUserIndexes[user][poolId] = 0 (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1242)
    - _burnFromInternalBalance(_poolAddr,msg.sender,LPTokensToBurn,true,true) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#946)
        - poolStates[token].totalSupplyBase.amount -= (amount).toUint224() (contracts/mosaic-alpha-contracts/Vault/Vault.sol#531)
        - poolStates[token].totalSupplyBase.timestamp = uint32(block.timestamp) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#532)
        - poolStates[_pool].lastTrailingTimestamp = uint48(block.timestamp) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#560)
        - poolStates[_pool].totalSupplyBase.amount += uint224(expAmount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#603)
        - poolStates[_pool].totalSupplyBase.timestamp = (block.timestamp).toUint32() (contracts/mosaic-alpha-contracts/Vault/Vault.sol#604)
        - poolStates[_pool].lastPerformanceTimestamp = uint48(block.timestamp) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#570)
Reentrancy in Vault.Mint(uint32,uint256,uint256[],address,address,bool,uint256,uint256) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#841-904):
    External calls:
    - (amountsToSpend,buyFeeLpTokens,None) = IPool(_poolAddr)._mint(LPTokensRequested,poolStates[_poolAddr].totalSupplyBase) (contracts/mosaic-alpha-
contracts/Vault/Vault.sol#861)
    - _transferInternalBalance(tokens[i],msg.sender,_poolAddr,amountsToSpend[i]) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#872)
        - IMosaicOracle(mosaicOracle).consultUsdPoolPricePerLp(poolAddr,ts) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#703)
        - IFees(fees).onBeforeTransfer(_pool,_from,_to,_internalBalance[_from][_pool],_internalBalance[_to]
[_pool],amount,originalTotalsupplyBase,trader,payload) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#598-601)
        - IPool(token).emitTransfer(from,to,amount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#499)
        - performanceFeesToMint = (IPool(_pool).updatePerfFeePricePerLp(tsTmp,valuePerLp)) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#568)
        - IPool(_pool).emitTransfer(address(0),fees,amount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#610)
        - IFees(fees).onBeforeTransfer(_pool,_from,_to,_internalBalance[_from][_pool],_internalBalance[_to][_pool],amount,0,address(0),payload)
(contracts/mosaic-alpha-contracts/Vault/Vault.sol#614)
    State variables written after the call(s):
    - _transferInternalBalance(tokens[i],msg.sender,_poolAddr,amountsToSpend[i]) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#872)
        - poolStates[_pool].lastTrailingTimestamp = uint48(block.timestamp) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#560)
        - poolStates[_pool].totalSupplyBase.amount += uint224(expAmount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#603)
        - poolStates[_pool].totalSupplyBase.timestamp = (block.timestamp).toUint32() (contracts/mosaic-alpha-contracts/Vault/Vault.sol#604)
        - poolStates[_pool].lastPerformanceTimestamp = uint48(block.timestamp) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#570)
Reentrancy in Vault.Mint(uint32,uint256,uint256[],address,address,bool,uint256,uint256) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#841-904):
    External calls:
    - (amountsToSpend,buyFeeLpTokens,None) = IPool(_poolAddr)._mint(LPTokensRequested,poolStates[_poolAddr].totalSupplyBase) (contracts/mosaic-alpha-
contracts/Vault/Vault.sol#861)
    - _registerUser(to,referredBy) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#886)
        - IUserProfile(userProfile).registerUser(to,referredBy) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#202)
    - _mintToInternalBalance(_poolAddr,to,LPTokensRequested,true,true) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#887)
        - IMosaicOracle(mosaicOracle).consultUsdPoolPricePerLp(poolAddr,ts) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#703)
        - IPool(token).emitTransfer(address(0),to,amount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#521)
        - IFees(fees).onBeforeTransfer(_pool,_from,_to,_internalBalance[_from][_pool],_internalBalance[_to]
[_pool],amount,originalTotalsupplyBase,trader,payload) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#598-601)
        - performanceFeesToMint = (IPool(_pool).updatePerfFeePricePerLp(tsTmp,valuePerLp)) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#568)
        - IPool(_pool).emitTransfer(address(0),fees,amount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#610)
        - IFees(fees).onBeforeTransfer(_pool,_from,_to,_internalBalance[_from][_pool],_internalBalance[_to][_pool],amount,0,address(0),payload)
(contracts/mosaic-alpha-contracts/Vault/Vault.sol#614)
    State variables written after the call(s):
    - _mintToInternalBalance(_poolAddr,to,LPTokensRequested,true,true) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#887)
        - poolStates[token].totalSupplyBase.amount += (amount).toUint224() (contracts/mosaic-alpha-contracts/Vault/Vault.sol#514)
        - poolStates[token].totalSupplyBase.timestamp = uint32(block.timestamp) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#515)
        - poolStates[_pool].lastTrailingTimestamp = uint48(block.timestamp) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#560)
```

```
        - poolStates[_pool].totalSupplyBase.amount += uint224(expAmount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#603)
        - poolStates[_pool].totalSupplyBase.timestamp = (block.timestamp).toUint32() (contracts/mosaic-alpha-contracts/Vault/Vault.sol#604)
        - poolStates[_pool].lastPerformanceTimestamp = uint48(block.timestamp) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#570)
Reentrancy in Vault.Mint(uint32,uint256,uint256[],address,address,bool,uint256,uint256) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#841-904):
    External calls:
    - (amountsToSpend,buyFeeLpTokens,None) = IPool(_poolAddr)._mint(LPTokensRequested,poolStates[_poolAddr].totalSupplyBase) (contracts/mosaic-alpha-
contracts/Vault/Vault.sol#861)
    - _registerUserPurchase(to,referredBy,trader,tokens,amountsToSpend,usdValue) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#891)
        - IAffiliate(affiliate).registerUserPurchase(to,referredBy,trader,usdAmount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#207)
        - IAffiliate(affiliate).registerUserPurchaseAsTokens(to,referredBy,trader,tokens,tokenAmounts) (contracts/mosaic-alpha-
contracts/Vault/Vault.sol#209)
    - (trailing,performance) = _calculateTrailingAndPerformanceFee(_poolAddr,true,false) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#893)
        - IMosaicOracle(mosaicOracle).consultUsdPoolPricePerLp(poolAddr,ts) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#703)
        - performanceFeesToMint = (IPool(_pool).updatePerfFeePricePerLp(tsTmp,valuePerLp)) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#568)
    State variables written after the call(s):
    - (trailing,performance) = _calculateTrailingAndPerformanceFee(_poolAddr,true,false) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#893)
        - poolStates[_pool].lastTrailingTimestamp = uint48(block.timestamp) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#560)
        - poolStates[_pool].lastPerformanceTimestamp = uint48(block.timestamp) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#570)
Reentrancy in Vault.Mint(uint32,uint256,uint256[],address,address,bool,uint256,uint256) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#841-904):
    External calls:
    - (amountsToSpend,buyFeeLpTokens,None) = IPool(_poolAddr)._mint(LPTokensRequested,poolStates[_poolAddr].totalSupplyBase) (contracts/mosaic-alpha-
contracts/Vault/Vault.sol#861)
    - _registerUserPurchase(to,referredBy,trader,tokens,amountsToSpend,usdValue) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#891)
        - IAffiliate(affiliate).registerUserPurchase(to,referredBy,trader,usdAmount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#207)
        - IAffiliate(affiliate).registerUserPurchaseAsTokens(to,referredBy,trader,tokens,tokenAmounts) (contracts/mosaic-alpha-
contracts/Vault/Vault.sol#209)
    - (trailing,performance) = _calculateTrailingAndPerformanceFee(_poolAddr,true,false) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#893)
        - IMosaicOracle(mosaicOracle).consultUsdPoolPricePerLp(poolAddr,ts) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#703)
        - performanceFeesToMint = (IPool(_pool).updatePerfFeePricePerLp(tsTmp,valuePerLp)) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#568)
    - IFees(fees).allocateTrailingFee(_poolAddr,trader,trailing,poolStates[_poolAddr].totalSupplyBase.amount,tx.origin) (contracts/mosaic-alpha-
contracts/Vault/Vault.sol#894)
    - IFees(fees).allocatePerformanceFee(_poolAddr,trader,performance,poolStates[_poolAddr].totalSupplyBase.amount,tx.origin) (contracts/mosaic-alpha-
contracts/Vault/Vault.sol#895)
    - _mintToInternalBalance(_poolAddr,fees,trailing + performance + buyFeeLpTokens,true,true) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#896)
        - IMosaicOracle(mosaicOracle).consultUsdPoolPricePerLp(poolAddr,ts) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#703)
        - IPool(token).emitTransfer(address(0),to,amount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#521)
        - IFees(fees).onBeforeTransfer(_pool,_from,_to,_internalBalance[_from][_pool],_internalBalance[_to]
[_pool],amount,originalTotalsupplyBase,trader,payload) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#598-601)
        - performanceFeesToMint = (IPool(_pool).updatePerfFeePricePerLp(tsTmp,valuePerLp)) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#568)
        - IPool(_pool).emitTransfer(address(0),fees,amount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#610)
        - IFees(fees).onBeforeTransfer(_pool,_from,_to,_internalBalance[_from][_pool],_internalBalance[_to][_pool],amount,0,address(0),payload)
(contracts/mosaic-alpha-contracts/Vault/Vault.sol#614)
    State variables written after the call(s):
    - _mintToInternalBalance(_poolAddr,fees,trailing + performance + buyFeeLpTokens,true,true) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#896)
        - poolStates[token].totalSupplyBase.amount += (amount).toUint224() (contracts/mosaic-alpha-contracts/Vault/Vault.sol#514)
        - poolStates[token].totalSupplyBase.timestamp = uint32(block.timestamp) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#515)
        - poolStates[_pool].lastTrailingTimestamp = uint48(block.timestamp) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#560)
        - poolStates[_pool].totalSupplyBase.amount += uint224(expAmount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#603)
        - poolStates[_pool].totalSupplyBase.timestamp = (block.timestamp).toUint32() (contracts/mosaic-alpha-contracts/Vault/Vault.sol#604)
        - poolStates[_pool].lastPerformanceTimestamp = uint48(block.timestamp) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#570)
Reentrancy in Vault.Mint(uint32,uint256,uint256[],address,address,bool,uint256,uint256) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#841-904):
    External calls:
    - (amountsToSpend,buyFeeLpTokens,None) = IPool(_poolAddr)._mint(LPTokensRequested,poolStates[_poolAddr].totalSupplyBase) (contracts/mosaic-alpha-
contracts/Vault/Vault.sol#861)
    - _registerUserPurchase(to,referredBy,trader,tokens,amountsToSpend,usdValue) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#891)
        - IAffiliate(affiliate).registerUserPurchase(to,referredBy,trader,usdAmount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#207)
        - IAffiliate(affiliate).registerUserPurchaseAsTokens(to,referredBy,trader,tokens,tokenAmounts) (contracts/mosaic-alpha-
contracts/Vault/Vault.sol#209)
    - (trailing,performance) = _calculateTrailingAndPerformanceFee(_poolAddr,true,false) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#893)
        - IMosaicOracle(mosaicOracle).consultUsdPoolPricePerLp(poolAddr,ts) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#703)
        - performanceFeesToMint = (IPool(_pool).updatePerfFeePricePerLp(tsTmp,valuePerLp)) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#568)
    - IFees(fees).allocateTrailingFee(_poolAddr,trader,trailing,poolStates[_poolAddr].totalSupplyBase.amount,tx.origin) (contracts/mosaic-alpha-
contracts/Vault/Vault.sol#894)
    - IFees(fees).allocatePerformanceFee(_poolAddr,trader,performance,poolStates[_poolAddr].totalSupplyBase.amount,tx.origin) (contracts/mosaic-alpha-
contracts/Vault/Vault.sol#895)
    - _mintToInternalBalance(_poolAddr,fees,trailing + performance + buyFeeLpTokens,true,true) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#896)
        - IMosaicOracle(mosaicOracle).consultUsdPoolPricePerLp(poolAddr,ts) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#703)
        - IPool(token).emitTransfer(address(0),to,amount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#521)
        - IFees(fees).onBeforeTransfer(_pool,_from,_to,_internalBalance[_from][_pool],_internalBalance[_to]
[_pool],amount,originalTotalsupplyBase,trader,payload) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#598-601)
        - performanceFeesToMint = (IPool(_pool).updatePerfFeePricePerLp(tsTmp,valuePerLp)) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#568)
        - IPool(_pool).emitTransfer(address(0),fees,amount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#610)
        - IFees(fees).onBeforeTransfer(_pool,_from,_to,_internalBalance[_from][_pool],_internalBalance[_to][_pool],amount,0,address(0),payload)
(contracts/mosaic-alpha-contracts/Vault/Vault.sol#614)
    - IFees(fees).allocateBuyFee(_poolAddr,to,trader,buyFeeLpTokens) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#898)
    - _mintToInternalBalance(_poolAddr,to,LPTokensRequested,true,true) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#899)
        - IMosaicOracle(mosaicOracle).consultUsdPoolPricePerLp(poolAddr,ts) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#703)
        - IPool(token).emitTransfer(address(0),to,amount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#521)
        - IFees(fees).onBeforeTransfer(_pool,_from,_to,_internalBalance[_from][_pool],_internalBalance[_to]
[_pool],amount,originalTotalsupplyBase,trader,payload) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#598-601)
        - performanceFeesToMint = (IPool(_pool).updatePerfFeePricePerLp(tsTmp,valuePerLp)) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#568)
        - IPool(_pool).emitTransfer(address(0),fees,amount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#610)
        - IFees(fees).onBeforeTransfer(_pool,_from,_to,_internalBalance[_from][_pool],_internalBalance[_to][_pool],amount,0,address(0),payload)
(contracts/mosaic-alpha-contracts/Vault/Vault.sol#614)
    State variables written after the call(s):
    - _mintToInternalBalance(_poolAddr,to,LPTokensRequested,true,true) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#899)
        - _internalBalance[to][token] += amount (contracts/mosaic-alpha-contracts/Vault/Vault.sol#517)
        - _internalBalance[fees][_pool] += expAmount (contracts/mosaic-alpha-contracts/Vault/Vault.sol#606)
    - _mintToInternalBalance(_poolAddr,to,LPTokensRequested,true,true) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#899)
        - poolOfUser[user].push(poolId) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1226)
        - poolOfUser[user][index_scope_0] = replacementPoolId (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1240)
        - poolOfUser[user].pop() (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1241)
    - _mintToInternalBalance(_poolAddr,to,LPTokensRequested,true,true) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#899)
        - poolOfUserIndexes[user][poolId] = (poolOfUser[user].length).toUint32() (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1228)
        - poolOfUserIndexes[user][replacementPoolId] = uint32(index_scope_0) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1239)
        - poolOfUserIndexes[user][poolId] = 0 (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1242)
    - _mintToInternalBalance(_poolAddr,to,LPTokensRequested,true,true) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#899)
        - poolStates[token].totalSupplyBase.amount += (amount).toUint224() (contracts/mosaic-alpha-contracts/Vault/Vault.sol#514)
        - poolStates[token].totalSupplyBase.timestamp = uint32(block.timestamp) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#515)
        - poolStates[_pool].lastTrailingTimestamp = uint48(block.timestamp) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#560)
        - poolStates[_pool].totalSupplyBase.amount += uint224(expAmount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#603)
        - poolStates[_pool].totalSupplyBase.timestamp = (block.timestamp).toUint32() (contracts/mosaic-alpha-contracts/Vault/Vault.sol#604)
        - poolStates[_pool].lastPerformanceTimestamp = uint48(block.timestamp) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#570)
Reentrancy in Vault._burnFromInternalBalance(address,address,uint256,bool,bool) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#526-537):
    External calls:
    - _onBeforeBalanceTransfer(token,from,address(0),amount,feesAlreadyAllocated,true,false) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#529)
        - IMosaicOracle(mosaicOracle).consultUsdPoolPricePerLp(poolAddr,ts) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#703)
        - IFees(fees).onBeforeTransfer(_pool,_from,_to,_internalBalance[_from][_pool],_internalBalance[_to]
[_pool],amount,originalTotalsupplyBase,trader,payload) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#598-601)
        - performanceFeesToMint = (IPool(_pool).updatePerfFeePricePerLp(tsTmp,valuePerLp)) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#568)
        - IPool(_pool).emitTransfer(address(0),fees,amount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#610)
        - IFees(fees).onBeforeTransfer(_pool,_from,_to,_internalBalance[_from][_pool],_internalBalance[_to][_pool],amount,0,address(0),payload)
(contracts/mosaic-alpha-contracts/Vault/Vault.sol#614)
    State variables written after the call(s):
    - _internalBalance[from][token] -= amount (contracts/mosaic-alpha-contracts/Vault/Vault.sol#530)
    - _handlerForPoolOfUser(from,token,false) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#534)
        - poolOfUser[user].push(poolId) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1226)
```

```
        - poolOfUser[user][index_scope_0] = replacementPoolId (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1240)
        - poolOfUser[user].pop() (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1241)
    - _handlerForPoolOfUser(from,token,false) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#534)
        - poolOfUserIndexes[user][poolId] = (poolOfUser[user].length).toUint32() (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1228)
        - poolOfUserIndexes[user][replacementPoolId] = uint32(index_scope_0) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1239)
        - poolOfUserIndexes[user][poolId] = 0 (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1242)
    - poolStates[token].totalSupplyBase.amount -= (amount).toUint224() (contracts/mosaic-alpha-contracts/Vault/Vault.sol#531)
    - poolStates[token].totalSupplyBase.timestamp = uint32(block.timestamp) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#532)
Reentrancy in Vault._calculateTrailingAndPerformanceFee(address,bool,bool) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#544-572):
    External calls:
    - valuePerLp = _consultPoolPricePerLpByAddr(_pool,tsTmp) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#566)
        - IMosaicOracle(mosaicOracle).consultUsdPoolPricePerLp(poolAddr,ts) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#703)
    - performanceFeesToMint = (IPool(_pool).updatePerfFeePricePerLp(tsTmp,valuePerLp)) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#568)
    State variables written after the call(s):
    - poolStates[_pool].lastPerformanceTimestamp = uint48(block.timestamp) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#570)
Reentrancy in Vault._mintToInternalBalance(address,address,uint256,bool,bool) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#510-522):
    External calls:
    - _onBeforeBalanceTransfer(token,address(0),to,amount,feesAlreadyAllocated,true,false) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#513)
        - IMosaicOracle(mosaicOracle).consultUsdPoolPricePerLp(poolAddr,ts) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#703)
        - IFees(fees).onBeforeTransfer(_pool,_from,_to,_internalBalance[_from][_pool],_internalBalance[_to]
[_pool],amount,originalTotalsupplyBase,trader,payload) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#598-601)
        - performanceFeesToMint = (IPool(_pool).updatePerfFeePricePerLp(tsTmp,valuePerLp)) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#568)
        - IPool(_pool).emitTransfer(address(0),fees,amount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#610)
        - IFees(fees).onBeforeTransfer(_pool,_from,_to,_internalBalance[_from][_pool],_internalBalance[_to][_pool],amount,0,address(0),payload)
(contracts/mosaic-alpha-contracts/Vault/Vault.sol#614)
    State variables written after the call(s):
    - _internalBalance[to][token] += amount (contracts/mosaic-alpha-contracts/Vault/Vault.sol#517)
    - _handlerForPoolOfUser(to,token,true) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#518)
        - poolOfUser[user].push(poolId) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1226)
        - poolOfUser[user][index_scope_0] = replacementPoolId (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1240)
        - poolOfUser[user].pop() (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1241)
    - _handlerForPoolOfUser(to,token,true) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#518)
        - poolOfUserIndexes[user][poolId] = (poolOfUser[user].length).toUint32() (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1228)
        - poolOfUserIndexes[user][replacementPoolId] = uint32(index_scope_0) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1239)
        - poolOfUserIndexes[user][poolId] = 0 (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1242)
    - poolStates[token].totalSupplyBase.amount += (amount).toUint224() (contracts/mosaic-alpha-contracts/Vault/Vault.sol#514)
    - poolStates[token].totalSupplyBase.timestamp = uint32(block.timestamp) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#515)
Reentrancy in Vault._onBeforeBalanceTransfer(address,address,address,uint256,bool,bool,bool) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#583-615):
    External calls:
    - (payload.trailingLpToMint,payload.performanceLpToMint) =
_calculateTrailingAndPerformanceFee(_pool,forceTrailingFeeAllocation,forcePerformanceFeeAllocation) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#591)
        - IMosaicOracle(mosaicOracle).consultUsdPoolPricePerLp(poolAddr,ts) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#703)
        - performanceFeesToMint = (IPool(_pool).updatePerfFeePricePerLp(tsTmp,valuePerLp)) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#568)
    - IFees(fees).onBeforeTransfer(_pool,_from,_to,_internalBalance[_from][_pool],_internalBalance[_to]
[_pool],amount,originalTotalsupplyBase,trader,payload) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#598-601)
    State variables written after the call(s):
    - _internalBalance[fees][_pool] += expAmount (contracts/mosaic-alpha-contracts/Vault/Vault.sol#606)
    - poolStates[_pool].totalSupplyBase.amount += uint224(expAmount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#603)
    - poolStates[_pool].totalSupplyBase.timestamp = (block.timestamp).toUint32() (contracts/mosaic-alpha-contracts/Vault/Vault.sol#604)
Reentrancy in Vault._transferInternalBalance(address,address,address,uint256,bool) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#485-506):
    External calls:
    - _onBeforeBalanceTransfer(token,from,to,amount,false,false,false) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#492)
        - IMosaicOracle(mosaicOracle).consultUsdPoolPricePerLp(poolAddr,ts) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#703)
        - IFees(fees).onBeforeTransfer(_pool,_from,_to,_internalBalance[_from][_pool],_internalBalance[_to]
[_pool],amount,originalTotalsupplyBase,trader,payload) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#598-601)
        - performanceFeesToMint = (IPool(_pool).updatePerfFeePricePerLp(tsTmp,valuePerLp)) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#568)
        - IPool(_pool).emitTransfer(address(0),fees,amount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#610)
        - IFees(fees).onBeforeTransfer(_pool,_from,_to,_internalBalance[_from][_pool],_internalBalance[_to][_pool],amount,0,address(0),payload)
(contracts/mosaic-alpha-contracts/Vault/Vault.sol#614)
    State variables written after the call(s):
    - _internalBalance[from][token] -= amount (contracts/mosaic-alpha-contracts/Vault/Vault.sol#494)
    - _internalBalance[to][token] += amount (contracts/mosaic-alpha-contracts/Vault/Vault.sol#495)
    - _handlerForPoolOfUser(from,token,false) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#496)
        - poolOfUser[user].push(poolId) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1226)
        - poolOfUser[user][index_scope_0] = replacementPoolId (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1240)
        - poolOfUser[user].pop() (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1241)
    - _handlerForPoolOfUser(to,token,true) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#497)
        - poolOfUser[user].push(poolId) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1226)
        - poolOfUser[user][index_scope_0] = replacementPoolId (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1240)
        - poolOfUser[user].pop() (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1241)
    - _handlerForPoolOfUser(from,token,false) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#496)
        - poolOfUserIndexes[user][poolId] = (poolOfUser[user].length).toUint32() (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1228)
        - poolOfUserIndexes[user][replacementPoolId] = uint32(index_scope_0) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1239)
        - poolOfUserIndexes[user][poolId] = 0 (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1242)
    - _handlerForPoolOfUser(to,token,true) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#497)
        - poolOfUserIndexes[user][poolId] = (poolOfUser[user].length).toUint32() (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1228)
        - poolOfUserIndexes[user][replacementPoolId] = uint32(index_scope_0) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1239)
        - poolOfUserIndexes[user][poolId] = 0 (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1242)
Reentrancy in Vault.addInitialLiquidity(uint32,address[],uint256[],address,bool) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#779-815):
    External calls:
    - (lpTokens,dust) = IPool(_poolAddress).initialLiquidityProvided(_tokens,_liquidity) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#806)
    - _mintToInternalBalance(_poolAddress,_lpTo,lpTokens,true,true) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#812)
        - IMosaicOracle(mosaicOracle).consultUsdPoolPricePerLp(poolAddr,ts) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#703)
        - IPool(token).emitTransfer(address(0),to,amount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#521)
        - IFees(fees).onBeforeTransfer(_pool,_from,_to,_internalBalance[_from][_pool],_internalBalance[_to]
[_pool],amount,originalTotalsupplyBase,trader,payload) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#598-601)
        - performanceFeesToMint = (IPool(_pool).updatePerfFeePricePerLp(tsTmp,valuePerLp)) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#568)
        - IPool(_pool).emitTransfer(address(0),fees,amount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#610)
        - IFees(fees).onBeforeTransfer(_pool,_from,_to,_internalBalance[_from][_pool],_internalBalance[_to][_pool],amount,0,address(0),payload)
(contracts/mosaic-alpha-contracts/Vault/Vault.sol#614)
    State variables written after the call(s):
    - _mintToInternalBalance(_poolAddress,_lpTo,lpTokens,true,true) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#812)
        - poolStates[token].totalSupplyBase.amount += (amount).toUint224() (contracts/mosaic-alpha-contracts/Vault/Vault.sol#514)
        - poolStates[token].totalSupplyBase.timestamp = uint32(block.timestamp) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#515)
        - poolStates[_pool].lastTrailingTimestamp = uint48(block.timestamp) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#560)
        - poolStates[_pool].totalSupplyBase.amount += uint224(expAmount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#603)
        - poolStates[_pool].totalSupplyBase.timestamp = (block.timestamp).toUint32() (contracts/mosaic-alpha-contracts/Vault/Vault.sol#604)
        - poolStates[_pool].lastPerformanceTimestamp = uint48(block.timestamp) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#570)
Reentrancy in Vault.addInitialLiquidity(uint32,address[],uint256[],address,bool) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#779-815):
    External calls:
    - (lpTokens,dust) = IPool(_poolAddress).initialLiquidityProvided(_tokens,_liquidity) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#806)
    - _mintToInternalBalance(_poolAddress,_lpTo,lpTokens,true,true) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#812)
        - IMosaicOracle(mosaicOracle).consultUsdPoolPricePerLp(poolAddr,ts) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#703)
        - IPool(token).emitTransfer(address(0),to,amount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#521)
        - IFees(fees).onBeforeTransfer(_pool,_from,_to,_internalBalance[_from][_pool],_internalBalance[_to]
[_pool],amount,originalTotalsupplyBase,trader,payload) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#598-601)
        - performanceFeesToMint = (IPool(_pool).updatePerfFeePricePerLp(tsTmp,valuePerLp)) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#568)
        - IPool(_pool).emitTransfer(address(0),fees,amount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#610)
        - IFees(fees).onBeforeTransfer(_pool,_from,_to,_internalBalance[_from][_pool],_internalBalance[_to][_pool],amount,0,address(0),payload)
(contracts/mosaic-alpha-contracts/Vault/Vault.sol#614)
    - _mintToInternalBalance(_poolAddress,address(0),dust,true,true) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#814)
        - IMosaicOracle(mosaicOracle).consultUsdPoolPricePerLp(poolAddr,ts) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#703)
        - IPool(token).emitTransfer(address(0),to,amount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#521)
        - IFees(fees).onBeforeTransfer(_pool,_from,_to,_internalBalance[_from][_pool],_internalBalance[_to]
[_pool],amount,originalTotalsupplyBase,trader,payload) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#598-601)
        - performanceFeesToMint = (IPool(_pool).updatePerfFeePricePerLp(tsTmp,valuePerLp)) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#568)
        - IPool(_pool).emitTransfer(address(0),fees,amount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#610)
```

```
        - IFees(fees).onBeforeTransfer(_pool,_from,_to,_internalBalance[_from][_pool],_internalBalance[_to][_pool],amount,0,address(0),payload)
(contracts/mosaic-alpha-contracts/Vault/Vault.sol#614)
    State variables written after the call(s):
    - _mintToInternalBalance(_poolAddress,address(0),dust,true,true) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#814)
        - _internalBalance[to][token] += amount (contracts/mosaic-alpha-contracts/Vault/Vault.sol#517)
        - _internalBalance[fees][_pool] += expAmount (contracts/mosaic-alpha-contracts/Vault/Vault.sol#606)
    - _mintToInternalBalance(_poolAddress,address(0),dust,true,true) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#814)
        - poolOfUser[user].push(poolId) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1226)
        - poolOfUser[user][index_scope_0] = replacementPoolId (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1240)
        - poolOfUser[user].pop() (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1241)
    - _mintToInternalBalance(_poolAddress,address(0),dust,true,true) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#814)
        - poolOfUserIndexes[user][poolId] = (poolOfUser[user].length).toUint32() (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1228)
        - poolOfUserIndexes[user][replacementPoolId] = uint32(index_scope_0) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1239)
        - poolOfUserIndexes[user][poolId] = 0 (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1242)
    - _mintToInternalBalance(_poolAddress,address(0),dust,true,true) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#814)
        - poolStates[token].totalSupplyBase.amount += (amount).toUint224() (contracts/mosaic-alpha-contracts/Vault/Vault.sol#514)
        - poolStates[token].totalSupplyBase.timestamp = uint32(block.timestamp) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#515)
        - poolStates[_pool].lastTrailingTimestamp = uint48(block.timestamp) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#560)
        - poolStates[_pool].totalSupplyBase.amount += uint224(expAmount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#603)
        - poolStates[_pool].totalSupplyBase.timestamp = (block.timestamp).toUint32() (contracts/mosaic-alpha-contracts/Vault/Vault.sol#604)
        - poolStates[_pool].lastPerformanceTimestamp = uint48(block.timestamp) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#570)
Reentrancy in Vault.bid(uint256,uint256,bool,bool,uint256) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1163-1213):
    External calls:
    - _transferInternalBalance(_auctionDetails.tokenToBuy,msg.sender,_auctionDetails.poolAddress,_toPay) (contracts/mosaic-alpha-
contracts/Vault/Vault.sol#1199)
        - IMosaicOracle(mosaicOracle).consultUsdPoolPricePerLp(poolAddr,ts) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#703)
        - IFees(fees).onBeforeTransfer(_pool,_from,_to,_internalBalance[_from][_pool],_internalBalance[_to]
[_pool],amount,originalTotalsupplyBase,trader,payload) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#598-601)
        - IPool(token).emitTransfer(from,to,amount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#499)
        - performanceFeesToMint = (IPool(_pool).updatePerfFeePricePerLp(tsTmp,valuePerLp)) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#568)
        - IPool(_pool).emitTransfer(address(0),fees,amount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#610)
        - IFees(fees).onBeforeTransfer(_pool,_from,_to,_internalBalance[_from][_pool],_internalBalance[_to][_pool],amount,0,address(0),payload)
(contracts/mosaic-alpha-contracts/Vault/Vault.sol#614)
    - _transferInternalBalance(_auctionDetails.tokenToSell,_auctionDetails.poolAddress,msg.sender,amount) (contracts/mosaic-alpha-
contracts/Vault/Vault.sol#1201)
        - IMosaicOracle(mosaicOracle).consultUsdPoolPricePerLp(poolAddr,ts) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#703)
        - IFees(fees).onBeforeTransfer(_pool,_from,_to,_internalBalance[_from][_pool],_internalBalance[_to]
[_pool],amount,originalTotalsupplyBase,trader,payload) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#598-601)
        - IPool(token).emitTransfer(from,to,amount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#499)
        - performanceFeesToMint = (IPool(_pool).updatePerfFeePricePerLp(tsTmp,valuePerLp)) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#568)
        - IPool(_pool).emitTransfer(address(0),fees,amount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#610)
        - IFees(fees).onBeforeTransfer(_pool,_from,_to,_internalBalance[_from][_pool],_internalBalance[_to][_pool],amount,0,address(0),payload)
(contracts/mosaic-alpha-contracts/Vault/Vault.sol#614)
    State variables written after the call(s):
    - _transferInternalBalance(_auctionDetails.tokenToSell,_auctionDetails.poolAddress,msg.sender,amount) (contracts/mosaic-alpha-
contracts/Vault/Vault.sol#1201)
        - _internalBalance[from][token] -= amount (contracts/mosaic-alpha-contracts/Vault/Vault.sol#494)
        - _internalBalance[to][token] += amount (contracts/mosaic-alpha-contracts/Vault/Vault.sol#495)
        - _internalBalance[fees][_pool] += expAmount (contracts/mosaic-alpha-contracts/Vault/Vault.sol#606)
        - _internalBalance[from][token] -= amount (contracts/mosaic-alpha-contracts/Vault/Vault.sol#502)
        - _internalBalance[to][token] += amount (contracts/mosaic-alpha-contracts/Vault/Vault.sol#503)
    - _transferInternalBalance(_auctionDetails.tokenToSell,_auctionDetails.poolAddress,msg.sender,amount) (contracts/mosaic-alpha-
contracts/Vault/Vault.sol#1201)
        - poolOfUser[user].push(poolId) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1226)
        - poolOfUser[user][index_scope_0] = replacementPoolId (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1240)
        - poolOfUser[user].pop() (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1241)
    - _transferInternalBalance(_auctionDetails.tokenToSell,_auctionDetails.poolAddress,msg.sender,amount) (contracts/mosaic-alpha-
contracts/Vault/Vault.sol#1201)
        - poolOfUserIndexes[user][poolId] = (poolOfUser[user].length).toUint32() (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1228)
        - poolOfUserIndexes[user][replacementPoolId] = uint32(index_scope_0) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1239)
        - poolOfUserIndexes[user][poolId] = 0 (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1242)
    - _transferInternalBalance(_auctionDetails.tokenToSell,_auctionDetails.poolAddress,msg.sender,amount) (contracts/mosaic-alpha-
contracts/Vault/Vault.sol#1201)
        - poolStates[_pool].lastTrailingTimestamp = uint48(block.timestamp) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#560)
        - poolStates[_pool].totalSupplyBase.amount += uint224(expAmount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#603)
        - poolStates[_pool].totalSupplyBase.timestamp = (block.timestamp).toUint32() (contracts/mosaic-alpha-contracts/Vault/Vault.sol#604)
        - poolStates[_pool].lastPerformanceTimestamp = uint48(block.timestamp) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#570)
Reentrancy in Vault.registerPool(address,address,address) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#721-739):
    External calls:
    - _registerUser(_user,_referredBy) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#727)
        - IUserProfile(userProfile).registerUser(to,referredBy) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#202)
    State variables written after the call(s):
    - _poolAddressToId[_pool] = _poolId (contracts/mosaic-alpha-contracts/Vault/Vault.sol#732)
Reentrancy in Vault.swap(address,bool,address,address,uint256,bool,uint256,uint64) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#969-1032):
    External calls:
    - _transferInternalBalance(tokenIn,msg.sender,poolAddress,_amountIn) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1010)
        - IMosaicOracle(mosaicOracle).consultUsdPoolPricePerLp(poolAddr,ts) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#703)
        - IFees(fees).onBeforeTransfer(_pool,_from,_to,_internalBalance[_from][_pool],_internalBalance[_to]
[_pool],amount,originalTotalsupplyBase,trader,payload) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#598-601)
        - IPool(token).emitTransfer(from,to,amount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#499)
        - performanceFeesToMint = (IPool(_pool).updatePerfFeePricePerLp(tsTmp,valuePerLp)) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#568)
        - IPool(_pool).emitTransfer(address(0),fees,amount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#610)
        - IFees(fees).onBeforeTransfer(_pool,_from,_to,_internalBalance[_from][_pool],_internalBalance[_to][_pool],amount,0,address(0),payload)
(contracts/mosaic-alpha-contracts/Vault/Vault.sol#614)
    - _transferInternalBalance(tokenOut,poolAddress,msg.sender,_amountOut) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1012)
        - IMosaicOracle(mosaicOracle).consultUsdPoolPricePerLp(poolAddr,ts) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#703)
        - IFees(fees).onBeforeTransfer(_pool,_from,_to,_internalBalance[_from][_pool],_internalBalance[_to]
[_pool],amount,originalTotalsupplyBase,trader,payload) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#598-601)
        - IPool(token).emitTransfer(from,to,amount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#499)
        - performanceFeesToMint = (IPool(_pool).updatePerfFeePricePerLp(tsTmp,valuePerLp)) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#568)
        - IPool(_pool).emitTransfer(address(0),fees,amount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#610)
        - IFees(fees).onBeforeTransfer(_pool,_from,_to,_internalBalance[_from][_pool],_internalBalance[_to][_pool],amount,0,address(0),payload)
(contracts/mosaic-alpha-contracts/Vault/Vault.sol#614)
    State variables written after the call(s):
    - _transferInternalBalance(tokenOut,poolAddress,msg.sender,_amountOut) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1012)
        - _internalBalance[from][token] -= amount (contracts/mosaic-alpha-contracts/Vault/Vault.sol#494)
        - _internalBalance[to][token] += amount (contracts/mosaic-alpha-contracts/Vault/Vault.sol#495)
        - _internalBalance[fees][_pool] += expAmount (contracts/mosaic-alpha-contracts/Vault/Vault.sol#606)
        - _internalBalance[from][token] -= amount (contracts/mosaic-alpha-contracts/Vault/Vault.sol#502)
        - _internalBalance[to][token] += amount (contracts/mosaic-alpha-contracts/Vault/Vault.sol#503)
    - _transferInternalBalance(tokenOut,poolAddress,msg.sender,_amountOut) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1012)
        - poolOfUser[user].push(poolId) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1226)
        - poolOfUser[user][index_scope_0] = replacementPoolId (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1240)
        - poolOfUser[user].pop() (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1241)
    - _transferInternalBalance(tokenOut,poolAddress,msg.sender,_amountOut) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1012)
        - poolOfUserIndexes[user][poolId] = (poolOfUser[user].length).toUint32() (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1228)
        - poolOfUserIndexes[user][replacementPoolId] = uint32(index_scope_0) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1239)
        - poolOfUserIndexes[user][poolId] = 0 (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1242)
    - _transferInternalBalance(tokenOut,poolAddress,msg.sender,_amountOut) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1012)
        - poolStates[_pool].lastTrailingTimestamp = uint48(block.timestamp) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#560)
        - poolStates[_pool].totalSupplyBase.amount += uint224(expAmount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#603)
        - poolStates[_pool].totalSupplyBase.timestamp = (block.timestamp).toUint32() (contracts/mosaic-alpha-contracts/Vault/Vault.sol#604)
        - poolStates[_pool].lastPerformanceTimestamp = uint48(block.timestamp) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#570)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1

Reentrancy in Fees._getParentsFromCache(address) (contracts/mosaic-alpha-contracts/Fees/Fees.sol#638-650):
    External calls:
```

```
    - IUserProfile(userProfile).registerUserWoBooster(_user) (contracts/mosaic-alpha-contracts/Fees/Fees.sol#642)
    State variables written after the call(s):
    - userFeeLevels[_user].parent = parent (contracts/mosaic-alpha-contracts/Fees/Fees.sol#644)
    - userFeeLevels[_user].parent2 = parent2 (contracts/mosaic-alpha-contracts/Fees/Fees.sol#645)
Reentrancy in Fees.allocateBuyFee(address,address,address,uint256) (contracts/mosaic-alpha-contracts/Fees/Fees.sol#417-446):
    External calls:
    - affRew = _getCappedAffiliateAmount(parent.user,uint128(affRew)) (contracts/mosaic-alpha-contracts/Fees/Fees.sol#436)
        -
IFeesClaimLimitAdapter(claimLimitAdapter).onBeforeAffiliatePayout(_user,_amount,userFeeLevels[_user].claimLimit,userFeeLevels[_user].claimLimitxTime,userFeeL
r].lastTime).toUint128() (contracts/mosaic-alpha-contracts/Fees/Fees.sol#200)
    - affRew2 = _getCappedAffiliateAmount(parent2.user,uint128(affRew2)) (contracts/mosaic-alpha-contracts/Fees/Fees.sol#437)
        -
IFeesClaimLimitAdapter(claimLimitAdapter).onBeforeAffiliatePayout(_user,_amount,userFeeLevels[_user].claimLimit,userFeeLevels[_user].claimLimitxTime,userFeeL
r].lastTime).toUint128() (contracts/mosaic-alpha-contracts/Fees/Fees.sol#200)
    State variables written after the call(s):
    - userPoolFeeStatus[parent.user][_pool].userDirectlyClaimableFee += uint128(affRew) (contracts/mosaic-alpha-contracts/Fees/Fees.sol#441)
    - userPoolFeeStatus[parent2.user][_pool].userDirectlyClaimableFee += uint128(affRew2) (contracts/mosaic-alpha-contracts/Fees/Fees.sol#442)
    - userPoolFeeStatus[feeTo][_pool].userDirectlyClaimableFee += uint128(_buyFeeLpTokens - feeDist.userRebateAmount - feeDist.traderAmount - affRew -
affRew2) (contracts/mosaic-alpha-contracts/Fees/Fees.sol#445)
Reentrancy in Fees.claimTraderClaimableFee(address) (contracts/mosaic-alpha-contracts/Fees/Fees.sol#308-355):
    External calls:
    - affRew = _getCappedAffiliateAmount(parent.user,uint128(affRew)) (contracts/mosaic-alpha-contracts/Fees/Fees.sol#333)
        -
IFeesClaimLimitAdapter(claimLimitAdapter).onBeforeAffiliatePayout(_user,_amount,userFeeLevels[_user].claimLimit,userFeeLevels[_user].claimLimitxTime,userFeeL
r].lastTime).toUint128() (contracts/mosaic-alpha-contracts/Fees/Fees.sol#200)
    - affRew2 = _getCappedAffiliateAmount(parent2.user,uint128(affRew2)) (contracts/mosaic-alpha-contracts/Fees/Fees.sol#334)
        -
IFeesClaimLimitAdapter(claimLimitAdapter).onBeforeAffiliatePayout(_user,_amount,userFeeLevels[_user].claimLimit,userFeeLevels[_user].claimLimitxTime,userFeeL
r].lastTime).toUint128() (contracts/mosaic-alpha-contracts/Fees/Fees.sol#200)
    State variables written after the call(s):
    - userPoolFeeStatus[parent.user][_pool].userDirectlyClaimableFee += uint128(affRew) (contracts/mosaic-alpha-contracts/Fees/Fees.sol#336)
    - userPoolFeeStatus[parent2.user][_pool].userDirectlyClaimableFee += uint128(affRew2) (contracts/mosaic-alpha-contracts/Fees/Fees.sol#337)
    - userPoolFeeStatus[feeTo][_pool].userDirectlyClaimableFee += uint128(lostTraderClaimableVariableFee + affiliateAmount - affRew - affRew2)
(contracts/mosaic-alpha-contracts/Fees/Fees.sol#343)
Reentrancy in Fees.claimTraderClaimableFee(address) (contracts/mosaic-alpha-contracts/Fees/Fees.sol#308-355):
    External calls:
    - affRew = _getCappedAffiliateAmount(parent.user,uint128(affRew)) (contracts/mosaic-alpha-contracts/Fees/Fees.sol#333)
        -
IFeesClaimLimitAdapter(claimLimitAdapter).onBeforeAffiliatePayout(_user,_amount,userFeeLevels[_user].claimLimit,userFeeLevels[_user].claimLimitxTime,userFeeL
r].lastTime).toUint128() (contracts/mosaic-alpha-contracts/Fees/Fees.sol#200)
    - affRew2 = _getCappedAffiliateAmount(parent2.user,uint128(affRew2)) (contracts/mosaic-alpha-contracts/Fees/Fees.sol#334)
        -
IFeesClaimLimitAdapter(claimLimitAdapter).onBeforeAffiliatePayout(_user,_amount,userFeeLevels[_user].claimLimit,userFeeLevels[_user].claimLimitxTime,userFeeL
r].lastTime).toUint128() (contracts/mosaic-alpha-contracts/Fees/Fees.sol#200)
    - IERC20(_pool).safeTransfer(_trader,amount - affiliateAmount) (contracts/mosaic-alpha-contracts/Fees/Fees.sol#349)
    State variables written after the call(s):
    - userPoolFeeStatus[executor][_pool].userDirectlyClaimableFee += uint128(executorReward) (contracts/mosaic-alpha-contracts/Fees/Fees.sol#352)
Reentrancy in Fees.claimUserClaimableFee(address,address) (contracts/mosaic-alpha-contracts/Fees/Fees.sol#205-305):
    External calls:
    - toPay += _getCappedAffiliateAmount(_user,affiToPay) (contracts/mosaic-alpha-contracts/Fees/Fees.sol#278)
        -
IFeesClaimLimitAdapter(claimLimitAdapter).onBeforeAffiliatePayout(_user,_amount,userFeeLevels[_user].claimLimit,userFeeLevels[_user].claimLimitxTime,userFeeL
r].lastTime).toUint128() (contracts/mosaic-alpha-contracts/Fees/Fees.sol#200)
    State variables written after the call(s):
    - userPoolFeeStatus[executor][_pool].userDirectlyClaimableFee += uint128(executorReward) (contracts/mosaic-alpha-contracts/Fees/Fees.sol#291)
    - userPoolFeeStatus[feeTo][_pool].userDirectlyClaimableFee += uint128(totalClaimable) - toPay - executorReward (contracts/mosaic-alpha-
contracts/Fees/Fees.sol#294)
    - userPoolFeeStatus[_user][_pool].userDirectlyClaimableFee = 0 (contracts/mosaic-alpha-contracts/Fees/Fees.sol#300)
Reentrancy in Fees.onBeforeTransfer(address,address,address,uint256,uint256,uint256,uint256,address,IFees.OnBeforeTransferPayload) (contracts/mosaic-alpha-
contracts/Fees/Fees.sol#537-636):
    External calls:
    - (parentFrom,parentFrom2) = _getParentsFromCache(_from) (contracts/mosaic-alpha-contracts/Fees/Fees.sol#590)
        - IUserProfile(userProfile).registerUserWoBooster(_user) (contracts/mosaic-alpha-contracts/Fees/Fees.sol#642)
    State variables written after the call(s):
    - _allocateUserL1ClaimableFees(_pool,parentFrom) (contracts/mosaic-alpha-contracts/Fees/Fees.sol#597)
        - userPoolFeeStatus[_user][_pool].userClaimableL1Fee += uint128((l1Base * (claimablePerLp - claimablePerLpBefore)) >>
CLAIMABLE_PER_LP_MULTIPLIER_EXPONENT) (contracts/mosaic-alpha-contracts/Fees/Fees.sol#390)
        - userPoolFeeStatus[_user][_pool].lastClaimableAffiliateL1FeePerLp = claimablePerLp (contracts/mosaic-alpha-contracts/Fees/Fees.sol#391)
    - _allocateUserL2ClaimableFees(_pool,parentFrom2) (contracts/mosaic-alpha-contracts/Fees/Fees.sol#598)
        - userPoolFeeStatus[_user][_pool].userClaimableL2Fee += uint128((l2Base * (claimablePerLp - claimablePerLpBefore)) >>
CLAIMABLE_PER_LP_MULTIPLIER_EXPONENT) (contracts/mosaic-alpha-contracts/Fees/Fees.sol#403)
        - userPoolFeeStatus[_user][_pool].lastClaimableAffiliateL2FeePerLp = claimablePerLp (contracts/mosaic-alpha-contracts/Fees/Fees.sol#404)
    - _allocateUserClaimableFees(_pool,_to,_toBalanceBefore) (contracts/mosaic-alpha-contracts/Fees/Fees.sol#608)
        - userPoolFeeStatus[_user][_pool].userClaimableFee += uint128((oldBalance * (claimablePerLp - claimablePerLpBefore)) >>
CLAIMABLE_PER_LP_MULTIPLIER_EXPONENT) (contracts/mosaic-alpha-contracts/Fees/Fees.sol#377)
        - userPoolFeeStatus[_user][_pool].lastClaimableUserFeePerLp = claimablePerLp (contracts/mosaic-alpha-contracts/Fees/Fees.sol#378)
Reentrancy in Fees.onBeforeTransfer(address,address,address,uint256,uint256,uint256,uint256,address,IFees.OnBeforeTransferPayload) (contracts/mosaic-alpha-
contracts/Fees/Fees.sol#537-636):
    External calls:
    - (parentFrom,parentFrom2) = _getParentsFromCache(_from) (contracts/mosaic-alpha-contracts/Fees/Fees.sol#590)
        - IUserProfile(userProfile).registerUserWoBooster(_user) (contracts/mosaic-alpha-contracts/Fees/Fees.sol#642)
    - (parentTo,parentTo2) = _getParentsFromCache(_to) (contracts/mosaic-alpha-contracts/Fees/Fees.sol#617)
        - IUserProfile(userProfile).registerUserWoBooster(_user) (contracts/mosaic-alpha-contracts/Fees/Fees.sol#642)
    State variables written after the call(s):
    - (parentTo,parentTo2) = _getParentsFromCache(_to) (contracts/mosaic-alpha-contracts/Fees/Fees.sol#617)
        - userFeeLevels[_user].parentsCached = true (contracts/mosaic-alpha-contracts/Fees/Fees.sol#640)
        - userFeeLevels[_user].parent = parent (contracts/mosaic-alpha-contracts/Fees/Fees.sol#644)
        - userFeeLevels[_user].parent2 = parent2 (contracts/mosaic-alpha-contracts/Fees/Fees.sol#645)
    - _allocateUserL1ClaimableFees(_pool,parentTo) (contracts/mosaic-alpha-contracts/Fees/Fees.sol#621)
        - userPoolFeeStatus[_user][_pool].userClaimableL1Fee += uint128((l1Base * (claimablePerLp - claimablePerLpBefore)) >>
CLAIMABLE_PER_LP_MULTIPLIER_EXPONENT) (contracts/mosaic-alpha-contracts/Fees/Fees.sol#390)
        - userPoolFeeStatus[_user][_pool].lastClaimableAffiliateL1FeePerLp = claimablePerLp (contracts/mosaic-alpha-contracts/Fees/Fees.sol#391)
    - _allocateUserL2ClaimableFees(_pool,parentTo2) (contracts/mosaic-alpha-contracts/Fees/Fees.sol#622)
        - userPoolFeeStatus[_user][_pool].userClaimableL2Fee += uint128((l2Base * (claimablePerLp - claimablePerLpBefore)) >>
CLAIMABLE_PER_LP_MULTIPLIER_EXPONENT) (contracts/mosaic-alpha-contracts/Fees/Fees.sol#403)
        - userPoolFeeStatus[_user][_pool].lastClaimableAffiliateL2FeePerLp = claimablePerLp (contracts/mosaic-alpha-contracts/Fees/Fees.sol#404)
    - userPoolFeeStatus[parentFrom][_pool].l1Balance -= uint128(_amount) (contracts/mosaic-alpha-contracts/Fees/Fees.sol#628)
    - userPoolFeeStatus[parentTo][_pool].l1Balance += uint128(_amount) (contracts/mosaic-alpha-contracts/Fees/Fees.sol#629)
    - userPoolFeeStatus[parentFrom2][_pool].l2Balance -= uint128(_amount) (contracts/mosaic-alpha-contracts/Fees/Fees.sol#632)
    - userPoolFeeStatus[parentTo2][_pool].l2Balance += uint128(_amount) (contracts/mosaic-alpha-contracts/Fees/Fees.sol#633)
Reentrancy in Fees.selfManageMe() (contracts/mosaic-alpha-contracts/Fees/Fees.sol#111-123):
    External calls:
    - _selfManageMeBefore() (contracts/mosaic-alpha-contracts/Fees/Fees.sol#115)
        - IUserProfile(userProfile).registerUser(slots[1]) (contracts/mosaic-alpha-contracts/Fees/Fees.sol#148)
    State variables written after the call(s):
    - governanceState.running = nextRunning (contracts/mosaic-alpha-contracts/Fees/Fees.sol#119)
    - governanceState.managers = IGovernance(governAddress).read_core_managers() (contracts/mosaic-alpha-contracts/Fees/Fees.sol#120)
    - governanceState.governanceAddress = IGovernance(governAddress).read_core_govAddr() (contracts/mosaic-alpha-contracts/Fees/Fees.sol#121)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1

Reentrancy in UserProfile._registerUser(address,address) (contracts/mosaic-alpha-contracts/User/UserProfile.sol#226-247):
    External calls:
    - _referredBy = IAffiliateBooster(affiliateBoosterContract).pickNextReferral() (contracts/mosaic-alpha-contracts/User/UserProfile.sol#230)
    - _registerUser(_referredBy,address(0)) (contracts/mosaic-alpha-contracts/User/UserProfile.sol#236)
        - _referredBy = IAffiliateBooster(affiliateBoosterContract).pickNextReferral() (contracts/mosaic-alpha-contracts/User/UserProfile.sol#230)
    State variables written after the call(s):
    - userCount ++ (contracts/mosaic-alpha-contracts/User/UserProfile.sol#237)
```

```
    - userReferrals[_referredBy].push(_user) (contracts/mosaic-alpha-contracts/User/UserProfile.sol#244)
    - _registerUser(_referredBy,address(0)) (contracts/mosaic-alpha-contracts/User/UserProfile.sol#236)
        - users[_user].exists = true (contracts/mosaic-alpha-contracts/User/UserProfile.sol#238)
        - users[_user].referredBy = _referredBy (contracts/mosaic-alpha-contracts/User/UserProfile.sol#239)
        - users[_user].referredByBefore = users[_referredBy].referredBy (contracts/mosaic-alpha-contracts/User/UserProfile.sol#240)
        - users[_user].referredByRank = users[_referredBy].rank (contracts/mosaic-alpha-contracts/User/UserProfile.sol#241)
        - users[_user].buyFeeDiscount = IAffiliate(affiliateContract).getLevelDetails(users[_referredBy].rank).referralBuyFeeDiscount (contracts/mosaic-
alpha-contracts/User/UserProfile.sol#242)
        - users[_referredBy].referralCount ++ (contracts/mosaic-alpha-contracts/User/UserProfile.sol#243)
    - users[_user].exists = true (contracts/mosaic-alpha-contracts/User/UserProfile.sol#238)
    - users[_user].referredBy = _referredBy (contracts/mosaic-alpha-contracts/User/UserProfile.sol#239)
    - users[_user].referredByBefore = users[_referredBy].referredBy (contracts/mosaic-alpha-contracts/User/UserProfile.sol#240)
    - users[_user].referredByRank = users[_referredBy].rank (contracts/mosaic-alpha-contracts/User/UserProfile.sol#241)
    - users[_user].buyFeeDiscount = IAffiliate(affiliateContract).getLevelDetails(users[_referredBy].rank).referralBuyFeeDiscount (contracts/mosaic-alpha-
contracts/User/UserProfile.sol#242)
    - users[_referredBy].referralCount ++ (contracts/mosaic-alpha-contracts/User/UserProfile.sol#243)
Reentrancy in Affiliate._registerUserPurchase(address,address,address,uint256) (contracts/mosaic-alpha-contracts/User/Affiliate.sol#220-279):
    External calls:
    - IUserProfile(userProfile).registerUser(user,referredBy) (contracts/mosaic-alpha-contracts/User/Affiliate.sol#221)
    - IUserProfile(userProfile).registerUser(trader,address(0)) (contracts/mosaic-alpha-contracts/User/Affiliate.sol#223)
    - IUserProfile(userProfile).increaseActiveReferralCount(referredBy) (contracts/mosaic-alpha-contracts/User/Affiliate.sol#232)
    - IUserProfile(userProfile).setUserRank(referredBy,newRank) (contracts/mosaic-alpha-contracts/User/Affiliate.sol#247)
    State variables written after the call(s):
    - userData[referredBy].affiliateRevision = revision (contracts/mosaic-alpha-contracts/User/Affiliate.sol#248)
Reentrancy in Affiliate._registerUserPurchase(address,address,address,uint256) (contracts/mosaic-alpha-contracts/User/Affiliate.sol#220-279):
    External calls:
    - IUserProfile(userProfile).registerUser(user,referredBy) (contracts/mosaic-alpha-contracts/User/Affiliate.sol#221)
    - IUserProfile(userProfile).registerUser(trader,address(0)) (contracts/mosaic-alpha-contracts/User/Affiliate.sol#223)
    - IUserProfile(userProfile).increaseActiveReferralCount(referredBy) (contracts/mosaic-alpha-contracts/User/Affiliate.sol#232)
    - IUserProfile(userProfile).setUserRank(referredBy,newRank) (contracts/mosaic-alpha-contracts/User/Affiliate.sol#243)
    - IUserProfile(userProfile).setUserRank(referredBy,newRank) (contracts/mosaic-alpha-contracts/User/Affiliate.sol#247)
    - IUserProfile(userProfile).setUserRank(user,newRank) (contracts/mosaic-alpha-contracts/User/Affiliate.sol#260)
    State variables written after the call(s):
    - userData[user].affiliateRevision = revision (contracts/mosaic-alpha-contracts/User/Affiliate.sol#261)
Reentrancy in Affiliate._registerUserPurchase(address,address,address,uint256) (contracts/mosaic-alpha-contracts/User/Affiliate.sol#220-279):
    External calls:
    - IUserProfile(userProfile).registerUser(user,referredBy) (contracts/mosaic-alpha-contracts/User/Affiliate.sol#221)
    - IUserProfile(userProfile).registerUser(trader,address(0)) (contracts/mosaic-alpha-contracts/User/Affiliate.sol#223)
    - IUserProfile(userProfile).increaseActiveReferralCount(referredBy) (contracts/mosaic-alpha-contracts/User/Affiliate.sol#232)
    - IUserProfile(userProfile).setUserRank(referredBy,newRank) (contracts/mosaic-alpha-contracts/User/Affiliate.sol#243)
    - IUserProfile(userProfile).setUserRank(referredBy,newRank) (contracts/mosaic-alpha-contracts/User/Affiliate.sol#247)
    - IUserProfile(userProfile).setUserRank(user,newRank) (contracts/mosaic-alpha-contracts/User/Affiliate.sol#256)
    - IUserProfile(userProfile).setUserRank(user,newRank) (contracts/mosaic-alpha-contracts/User/Affiliate.sol#260)
    State variables written after the call(s):
    - userData[trader].traderPurchase += usdAmount (contracts/mosaic-alpha-contracts/User/Affiliate.sol#265)
Reentrancy in Affiliate._registerUserPurchase(address,address,address,uint256) (contracts/mosaic-alpha-contracts/User/Affiliate.sol#220-279):
    External calls:
    - IUserProfile(userProfile).registerUser(user,referredBy) (contracts/mosaic-alpha-contracts/User/Affiliate.sol#221)
    - IUserProfile(userProfile).registerUser(trader,address(0)) (contracts/mosaic-alpha-contracts/User/Affiliate.sol#223)
    - IUserProfile(userProfile).increaseActiveReferralCount(referredBy) (contracts/mosaic-alpha-contracts/User/Affiliate.sol#232)
    - IUserProfile(userProfile).setUserRank(referredBy,newRank) (contracts/mosaic-alpha-contracts/User/Affiliate.sol#243)
    - IUserProfile(userProfile).setUserRank(referredBy,newRank) (contracts/mosaic-alpha-contracts/User/Affiliate.sol#247)
    - IUserProfile(userProfile).setUserRank(user,newRank) (contracts/mosaic-alpha-contracts/User/Affiliate.sol#256)
    - IUserProfile(userProfile).setUserRank(user,newRank) (contracts/mosaic-alpha-contracts/User/Affiliate.sol#260)
    - IUserProfile(userProfile).setUserRank(trader,newRank) (contracts/mosaic-alpha-contracts/User/Affiliate.sol#275)
    State variables written after the call(s):
    - userData[trader].affiliateRevision = revision (contracts/mosaic-alpha-contracts/User/Affiliate.sol#276)
Reentrancy in Swaps.addTokens(address[]) (contracts/mosaic-alpha-contracts/Vault/Swaps.sol#29-38):
    External calls:
    - _approveToken(token,type()(uint256).max) (contracts/mosaic-alpha-contracts/Vault/Swaps.sol#33)
        - IERC20(_token).approve(vaultAddress,_amount) (contracts/mosaic-alpha-contracts/Vault/Swaps.sol#25)
        - IERC20(_token).approve(pancakeRouterAddress,_amount) (contracts/mosaic-alpha-contracts/Vault/Swaps.sol#26)
    State variables written after the call(s):
    - tokens[token] = true (contracts/mosaic-alpha-contracts/Vault/Swaps.sol#34)
Reentrancy in AffiliateBooster.allocateTicket(address,address,uint256,uint256) (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#340-343):
    External calls:
    - _allocateTicket(_a,_referredBy,_c,_price) (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#342)
        - userProfile.registerUser(_a,_referredBy) (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#353)
        - controller.ticketAllocated(_a,_c) (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#362)
    - closeEpochIfTimeUp() (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#340)
        - controller.epochClosed(state.currentEpoch - 1) (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#427)
    State variables written after the call(s):
    - _allocateTicket(_a,_referredBy,_c,_price) (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#342)
        - epochTickets[state.currentEpoch].push(_a) (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#355)
    - _allocateTicket(_a,_referredBy,_c,_price) (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#342)
        - epochs[state.currentEpoch].ticketsBought += uint64(_c) (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#358)
    - _allocateTicket(_a,_referredBy,_c,_price) (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#342)
        - state.totalPlayerWeight += state.epochWeights[0] (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#356)
        - state.totalTicketCount += uint64(_c) (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#360)
Reentrancy in SimpleDeposit.depositAndLock(uint256) (contracts/mosaic-alpha-contracts/Deposit/SimpleDeposit.sol#200-203):
    External calls:
    - _deposit(msg.sender,msg.sender,_amount) (contracts/mosaic-alpha-contracts/Deposit/SimpleDeposit.sol#201)
        - successTransfer = _stakingToken.transferFrom(_sender,address(this),_amount) (contracts/mosaic-alpha-contracts/Deposit/SimpleDeposit.sol#279-283)
    - _userLock(msg.sender,_amount,address(0)) (contracts/mosaic-alpha-contracts/Deposit/SimpleDeposit.sol#202)
        - IAffiliate(_affiliateContract).userStakeChanged(_user,_referredBy,_amount) (contracts/mosaic-alpha-contracts/Deposit/SimpleDeposit.sol#130)
        - IFees(_feesContract).userStakeChanged(_user,_amount) (contracts/mosaic-alpha-contracts/Deposit/SimpleDeposit.sol#133)
    State variables written after the call(s):
    - _userLock(msg.sender,_amount,address(0)) (contracts/mosaic-alpha-contracts/Deposit/SimpleDeposit.sol#202)
        - accountBalances[_user].balance -= _amount (contracts/mosaic-alpha-contracts/Deposit/SimpleDeposit.sol#322)
        - accountBalances[_user].lockedBalance += _amount (contracts/mosaic-alpha-contracts/Deposit/SimpleDeposit.sol#323)
        - accountBalances[_user].lastLockTimestamp = uint64(block.timestamp) (contracts/mosaic-alpha-contracts/Deposit/SimpleDeposit.sol#324)
Reentrancy in SimpleDeposit.depositAndLock(uint256,address) (contracts/mosaic-alpha-contracts/Deposit/SimpleDeposit.sol#205-208):
    External calls:
    - _deposit(msg.sender,msg.sender,_amount) (contracts/mosaic-alpha-contracts/Deposit/SimpleDeposit.sol#206)
        - successTransfer = _stakingToken.transferFrom(_sender,address(this),_amount) (contracts/mosaic-alpha-contracts/Deposit/SimpleDeposit.sol#279-283)
    - _userLock(msg.sender,_amount,_referredBy) (contracts/mosaic-alpha-contracts/Deposit/SimpleDeposit.sol#207)
        - IAffiliate(_affiliateContract).userStakeChanged(_user,_referredBy,_amount) (contracts/mosaic-alpha-contracts/Deposit/SimpleDeposit.sol#130)
        - IFees(_feesContract).userStakeChanged(_user,_amount) (contracts/mosaic-alpha-contracts/Deposit/SimpleDeposit.sol#133)
    State variables written after the call(s):
    - _userLock(msg.sender,_amount,_referredBy) (contracts/mosaic-alpha-contracts/Deposit/SimpleDeposit.sol#207)
        - accountBalances[_user].balance -= _amount (contracts/mosaic-alpha-contracts/Deposit/SimpleDeposit.sol#322)
        - accountBalances[_user].lockedBalance += _amount (contracts/mosaic-alpha-contracts/Deposit/SimpleDeposit.sol#323)
        - accountBalances[_user].lastLockTimestamp = uint64(block.timestamp) (contracts/mosaic-alpha-contracts/Deposit/SimpleDeposit.sol#324)
Reentrancy in AffiliateBooster.pickNextReferral() (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#372-410):
    External calls:
    - closeEpochIfTimeUp() (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#372)
        - controller.epochClosed(state.currentEpoch - 1) (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#427)
    State variables written after the call(s):
    - epochTickets[state.currentEpoch - i][seed % epochTickets[state.currentEpoch - i].length] = epochTickets[state.currentEpoch - i]
[epochTickets[state.currentEpoch - i].length - 1] (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#390)
    - epochTickets[state.currentEpoch - i].pop() (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#391)
    - epochs[state.currentEpoch].ticketsSpent += 1 (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#399)
    - state.totalPlayerWeight -= state.epochWeights[i] (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#392)
    - state.totalTicketsSpent += 1 (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#398)
Reentrancy in AffiliateBooster.pickNextReferral() (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#372-410):
    External calls:
    - controller.referralPicked(pickedAddress,false) (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#404)
```

```
    - _incrementEpoch() (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#407)
        - controller.epochClosed(state.currentEpoch - 1) (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#427)
    - closeEpochIfTimeUp() (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#372)
        - controller.epochClosed(state.currentEpoch - 1) (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#427)
    State variables written after the call(s):
    - _incrementEpoch() (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#407)
        - epochs[state.currentEpoch].epochStart = uint64(block.timestamp) (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#421)
    - _incrementEpoch() (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#407)
        - state.totalPlayerWeight = 0 (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#416)
        - state.currentEpoch ++ (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#418)
        - state.currentEpochStart = uint64(block.timestamp) (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#420)
        - i < state.epochWeights.length && i <= state.currentEpoch (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#422)
        - state.totalPlayerWeight += state.epochWeights[i] * epochTickets[state.currentEpoch - i].length (contracts/mosaic-alpha-
contracts/User/AffiliateBooster.sol#423)
Reentrancy in AffiliateBooster.purchaseTicket(uint256) (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#313-315):
    External calls:
    - purchaseTicket(address(0),_c) (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#314)
        - userProfile.registerUser(_a,_referredBy) (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#353)
        - require(bool,string)(IERC20Burnable(state.feeToken).transferFrom(msg.sender,address(this),_c * state.ticketPrice),TransferFrom failed)
(contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#324)
        - IERC20Burnable(state.feeToken).burn(_c * state.ticketPrice) (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#326)
        - IERC20Burnable(state.feeToken).transfer(state.feeTo,_c * state.ticketPrice) (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#328)
        - controller.epochClosed(state.currentEpoch - 1) (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#427)
        - controller.ticketAllocated(_a,_c) (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#362)
    - closeEpochIfTimeUp() (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#313)
        - controller.epochClosed(state.currentEpoch - 1) (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#427)
    State variables written after the call(s):
    - purchaseTicket(address(0),_c) (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#314)
        - epochTickets[state.currentEpoch].push(_a) (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#355)
    - purchaseTicket(address(0),_c) (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#314)
        - epochs[state.currentEpoch].epochStart = uint64(block.timestamp) (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#421)
        - epochs[state.currentEpoch].ticketsBought += uint64(_c) (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#358)
    - purchaseTicket(address(0),_c) (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#314)
        - state.totalPlayerWeight = 0 (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#416)
        - state.currentEpoch ++ (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#418)
        - state.currentEpochStart = uint64(block.timestamp) (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#420)
        - i < state.epochWeights.length && i <= state.currentEpoch (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#422)
        - state.totalPlayerWeight += state.epochWeights[i] * epochTickets[state.currentEpoch - i].length (contracts/mosaic-alpha-
contracts/User/AffiliateBooster.sol#423)
        - state.totalPlayerWeight += state.epochWeights[0] (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#356)
        - state.totalTicketCount += uint64(_c) (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#360)
Reentrancy in AffiliateBooster.purchaseTicket(address,uint256) (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#322-331):
    External calls:
    - require(bool,string)(IERC20Burnable(state.feeToken).transferFrom(msg.sender,address(this),_c * state.ticketPrice),TransferFrom failed)
(contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#324)
    - IERC20Burnable(state.feeToken).burn(_c * state.ticketPrice) (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#326)
    - IERC20Burnable(state.feeToken).transfer(state.feeTo,_c * state.ticketPrice) (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#328)
    - _allocateTicket(msg.sender,_referredBy,_c,state.ticketPrice) (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#330)
        - userProfile.registerUser(_a,_referredBy) (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#353)
        - controller.ticketAllocated(_a,_c) (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#362)
    - closeEpochIfTimeUp() (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#322)
        - controller.epochClosed(state.currentEpoch - 1) (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#427)
    State variables written after the call(s):
    - _allocateTicket(msg.sender,_referredBy,_c,state.ticketPrice) (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#330)
        - epochTickets[state.currentEpoch].push(_a) (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#355)
    - _allocateTicket(msg.sender,_referredBy,_c,state.ticketPrice) (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#330)
        - epochs[state.currentEpoch].ticketsBought += uint64(_c) (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#358)
    - _allocateTicket(msg.sender,_referredBy,_c,state.ticketPrice) (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#330)
        - state.totalPlayerWeight += state.epochWeights[0] (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#356)
        - state.totalTicketCount += uint64(_c) (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#360)
Reentrancy in Swaps.removeTokens(address[]) (contracts/mosaic-alpha-contracts/Vault/Swaps.sol#40-49):
    External calls:
    - _approveToken(token,0) (contracts/mosaic-alpha-contracts/Vault/Swaps.sol#44)
        - IERC20(_token).approve(vaultAddress,_amount) (contracts/mosaic-alpha-contracts/Vault/Swaps.sol#25)
        - IERC20(_token).approve(pancakeRouterAddress,_amount) (contracts/mosaic-alpha-contracts/Vault/Swaps.sol#26)
    State variables written after the call(s):
    - tokens[token] = false (contracts/mosaic-alpha-contracts/Vault/Swaps.sol#45)
Reentrancy in UserProfile.setUserRank(address,uint8) (contracts/mosaic-alpha-contracts/User/UserProfile.sol#116-121):
    External calls:
    - _registerUser(_user,address(0)) (contracts/mosaic-alpha-contracts/User/UserProfile.sol#118)
        - _referredBy = IAffiliateBooster(affiliateBoosterContract).pickNextReferral() (contracts/mosaic-alpha-contracts/User/UserProfile.sol#230)
    State variables written after the call(s):
    - users[_user].rank = _rank (contracts/mosaic-alpha-contracts/User/UserProfile.sol#119)
Reentrancy in SimpleDeposit.unlockAndWithdraw(uint256) (contracts/mosaic-alpha-contracts/Deposit/SimpleDeposit.sol#210-213):
    External calls:
    - _userUnlock(msg.sender,_amount) (contracts/mosaic-alpha-contracts/Deposit/SimpleDeposit.sol#211)
        - IAffiliate(_affiliateContract).userStakeChanged(_user,_referredBy,_amount) (contracts/mosaic-alpha-contracts/Deposit/SimpleDeposit.sol#130)
        - IFees(_feesContract).userStakeChanged(_user,_amount) (contracts/mosaic-alpha-contracts/Deposit/SimpleDeposit.sol#133)
    - _withdraw(msg.sender,_amount) (contracts/mosaic-alpha-contracts/Deposit/SimpleDeposit.sol#212)
        - successTransfer = _stakingToken.transfer(_user,_amount) (contracts/mosaic-alpha-contracts/Deposit/SimpleDeposit.sol#306)
    State variables written after the call(s):
    - _withdraw(msg.sender,_amount) (contracts/mosaic-alpha-contracts/Deposit/SimpleDeposit.sol#212)
        - accountBalances[msg.sender].balance = accountBalances[msg.sender].balance - _amount (contracts/mosaic-alpha-
contracts/Deposit/SimpleDeposit.sol#303)
Reentrancy in Affiliate.userStakeChanged(address,address,uint256) (contracts/mosaic-alpha-contracts/User/Affiliate.sol#163-191):
    External calls:
    - IUserProfile(userProfile).registerUser(user,referredBy) (contracts/mosaic-alpha-contracts/User/Affiliate.sol#164)
    - IUserProfile(userProfile).setUserRank(user,newRank_scope_0) (contracts/mosaic-alpha-contracts/User/Affiliate.sol#188)
    State variables written after the call(s):
    - userData[user].affiliateRevision = revision (contracts/mosaic-alpha-contracts/User/Affiliate.sol#189)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1
```

dangerous-usage-of-txorigin

```
Vault.registerPool(address,address,address) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#721-739) uses tx.origin for authorization: _user != tx.origin
(contracts/mosaic-alpha-contracts/Vault/Vault.sol#726)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-usage-of-txorigin
```

uninitialized-local-variables

```
PancakeRouter._swapSupportingFeeOnTransferTokens(address[],address).i (contracts/mosaic-alpha-contracts/helpers/pancake/PancakeRouter.sol#324) is a local
variable never initialized
PancakeLibrary.getAmountsOut(address,uint256,address[]).i (contracts/mosaic-alpha-contracts/helpers/pancake/libraries/PancakeLibrary.sol#69) is a local
variable never initialized
PancakeRouter._swap(uint256[],address[],address).i (contracts/mosaic-alpha-contracts/helpers/pancake/PancakeRouter.sol#215) is a local variable never
initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables

Pool.updateWeights(uint32,uint32[]).i (contracts/mosaic-alpha-contracts/Pool/Pool.sol#675) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables

Vault.multiSwap(address[],address[],uint256,bool,uint256,uint64).i (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1042) is a local variable never
initialized
Governance.execute_batch_ManageBytes(address[],string[],bytes[]).i (contracts/mosaic-alpha-contracts/Governance/Governance.sol#502) is a local variable
```

```
never initialized
Vault._onBeforeBalanceTransfer(address,address,address,uint256,bool,bool,bool).payload (contracts/mosaic-alpha-contracts/Vault/Vault.sol#585) is a local
variable never initialized
Vault._calculateTrailingAndPerformanceFee(address,bool,bool).mintPerformance (contracts/mosaic-alpha-contracts/Vault/Vault.sol#546) is a local variable
never initialized
Vault._calculateTrailingAndPerformanceFee(address,bool,bool).mintTrailing (contracts/mosaic-alpha-contracts/Vault/Vault.sol#545) is a local variable never
initialized
Governance.execute_batch_Manage(address[]).i (contracts/mosaic-alpha-contracts/Governance/Governance.sol#489) is a local variable never initialized
Governance.onlyManagers().ok (contracts/mosaic-alpha-contracts/Governance/Governance.sol#113) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables

Fees.claimUserClaimableFee(address,address).affiToPay (contracts/mosaic-alpha-contracts/Fees/Fees.sol#230) is a local variable never initialized
Fees._updateTraderShareCurve(uint256[],uint32[]).i (contracts/mosaic-alpha-contracts/Fees/Fees.sol#690) is a local variable never initialized
Fees.onBeforeTransfer(address,address,address,uint256,uint256,uint256,uint256,address,IFees.OnBeforeTransferPayload).parentFrom (contracts/mosaic-alpha-
contracts/Fees/Fees.sol#568) is a local variable never initialized
Fees._updateDiscountCurve(uint256[],uint32[]).i (contracts/mosaic-alpha-contracts/Fees/Fees.sol#668) is a local variable never initialized
Fees.onBeforeTransfer(address,address,address,uint256,uint256,uint256,uint256,address,IFees.OnBeforeTransferPayload).parentTo2 (contracts/mosaic-alpha-
contracts/Fees/Fees.sol#571) is a local variable never initialized
Fees.claimUserClaimableFee(address,address).toPay (contracts/mosaic-alpha-contracts/Fees/Fees.sol#229) is a local variable never initialized
Fees.onBeforeTransfer(address,address,address,uint256,uint256,uint256,uint256,address,IFees.OnBeforeTransferPayload).parentTo (contracts/mosaic-alpha-
contracts/Fees/Fees.sol#570) is a local variable never initialized
Fees.onBeforeTransfer(address,address,address,uint256,uint256,uint256,uint256,address,IFees.OnBeforeTransferPayload).parentFrom2 (contracts/mosaic-alpha-
contracts/Fees/Fees.sol#569) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables

Affiliate.addNextLevel(uint8,uint16,uint16,uint16,uint16,uint32,uint256,uint256,uint256,uint256,string).tmp (contracts/mosaic-alpha-
contracts/User/Affiliate.sol#101) is a local variable never initialized
AffiliateBooster.pickNextReferral().i (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#386) is a local variable never initialized
Swaps.addTokens(address[]).i (contracts/mosaic-alpha-contracts/Vault/Swaps.sol#30) is a local variable never initialized
Swaps.removeTokens(address[]).i (contracts/mosaic-alpha-contracts/Vault/Swaps.sol#41) is a local variable never initialized
Swaps.quoteBurn(address,address[][],uint256)._usdReceived (contracts/mosaic-alpha-contracts/Vault/Swaps.sol#129) is a local variable never initialized
Swaps.quoteMint(address,address[][],uint256)._usdNeeded (contracts/mosaic-alpha-contracts/Vault/Swaps.sol#112) is a local variable never initialized
QueryApi.getPool(uint32)._poolInfo (contracts/mosaic-alpha-contracts/Query/QueryApi.sol#26) is a local variable never initialized
AffiliateBooster._incrementEpoch().i (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#422) is a local variable never initialized
Swaps.quickQuoteMint(address,address,uint256)._usdNeeded (contracts/mosaic-alpha-contracts/Vault/Swaps.sol#73) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables
```

unused-return

```
PancakeRouter._addLiquidity(address,address,uint256,uint256,uint256,uint256) (contracts/mosaic-alpha-contracts/helpers/pancake/PancakeRouter.sol#35-62)
ignores return value by IPancakeFactory(factory).createPair(tokenA,tokenB) (contracts/mosaic-alpha-contracts/helpers/pancake/PancakeRouter.sol#45)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return

Pool._transfer(address,uint256) (contracts/mosaic-alpha-contracts/Pool/Pool.sol#289-293) ignores return value by
IVault(vaultAddress).transferFromAsTokenContract(msg.sender,_to,_value) (contracts/mosaic-alpha-contracts/Pool/Pool.sol#290)
Pool._transferFrom(address,address,uint256) (contracts/mosaic-alpha-contracts/Pool/Pool.sol#304-310) ignores return value by
IVault(vaultAddress).transferFromAsTokenContract(_from,_to,_value) (contracts/mosaic-alpha-contracts/Pool/Pool.sol#307)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return

TradingBot.TradingStartFromPancake(ITradingBot.TradingInput) (contracts/mosaic-alpha-bot/Interfaces/TradingBot.sol#85-111) ignores return value by
IERC20(data.path[0]).approve(address(_vault),depositAmount) (contracts/mosaic-alpha-bot/Interfaces/TradingBot.sol#109)
TradingBot.TradingStartFromMosaic(ITradingBot.TradingInput) (contracts/mosaic-alpha-bot/Interfaces/TradingBot.sol#113-141) ignores return value by
IERC20(data.path[lastIndex]).approve(address(_vault),depositAmount) (contracts/mosaic-alpha-bot/Interfaces/TradingBot.sol#139)
PoolFactory.create(string,string,uint8,address[],uint32[],uint256[],IFees.MosaicPoolFees,address,bool) (contracts/mosaic-alpha-
contracts/Pool/PoolFactory.sol#120-263) ignores return value by IERC20(tokens[i]).approve(vaultAddress,initialLiquidity[i]) (contracts/mosaic-alpha-
contracts/Pool/PoolFactory.sol#203)
Swaps._approveToken(address,uint256) (contracts/mosaic-alpha-contracts/Vault/Swaps.sol#24-27) ignores return value by
IERC20(_token).approve(vaultAddress,_amount) (contracts/mosaic-alpha-contracts/Vault/Swaps.sol#25)
Swaps._approveToken(address,uint256) (contracts/mosaic-alpha-contracts/Vault/Swaps.sol#24-27) ignores return value by
IERC20(_token).approve(pancakeRouterAddress,_amount) (contracts/mosaic-alpha-contracts/Vault/Swaps.sol#26)
Swaps.mint(address,address,address[][],uint256,uint256,address,uint256) (contracts/mosaic-alpha-contracts/Vault/Swaps.sol#144-189) ignores return value by
IPancakeRouter02(pancakeRouterAddress).swapTokensForExactTokens(amountOut,amountInMax,path,to,deadline) (contracts/mosaic-alpha-
contracts/Vault/Swaps.sol#165)
Swaps.mint(address,address,address[][],uint256,uint256,address,uint256) (contracts/mosaic-alpha-contracts/Vault/Swaps.sol#144-189) ignores return value by
IVault(vaultAddress).Mint(poolId,lpTokensRequested,_requestedAmounts,msg.sender,referredBy,true,deadline,(usdMaxSpend - remainingUSDT)) (contracts/mosaic-
alpha-contracts/Vault/Swaps.sol#183)
Swaps.mint(address,address,address[][],uint256,uint256,address,uint256) (contracts/mosaic-alpha-contracts/Vault/Swaps.sol#144-189) ignores return value by
IVault(vaultAddress).Mint(poolId,lpTokensRequested,_requestedAmounts,msg.sender,referredBy,true,deadline,0) (contracts/mosaic-alpha-
contracts/Vault/Swaps.sol#183)
Swaps.burn(address,address,address[][],uint256,uint256,uint256) (contracts/mosaic-alpha-contracts/Vault/Swaps.sol#191-220) ignores return value by
IPancakeRouter02(pancakeRouterAddress).swapExactTokensForTokens(amountIn,amountOutMin,path,address(this),deadline) (contracts/mosaic-alpha-
contracts/Vault/Swaps.sol#210)
Swaps.quickMint(address,address,uint256,uint256,address,uint256) (contracts/mosaic-alpha-contracts/Vault/Swaps.sol#222-255) ignores return value by
IPancakeRouter02(pancakeRouterAddress).swapTokensForExactTokens(amountOut,amountInMax,path,swapAddr,deadline) (contracts/mosaic-alpha-
contracts/Vault/Swaps.sol#240)
Swaps.quickMint(address,address,uint256,uint256,address,uint256) (contracts/mosaic-alpha-contracts/Vault/Swaps.sol#222-255) ignores return value by
IVault(vaultAddress).Mint(poolId,lpTokensRequested,_requestedAmounts,msg.sender,referredBy,true,deadline,(usdMaxSpend - remainingUSD)) (contracts/mosaic-
alpha-contracts/Vault/Swaps.sol#250)
Swaps.quickMint(address,address,uint256,uint256,address,uint256) (contracts/mosaic-alpha-contracts/Vault/Swaps.sol#222-255) ignores return value by
IVault(vaultAddress).Mint(poolId,lpTokensRequested,_requestedAmounts,msg.sender,referredBy,true,deadline,0) (contracts/mosaic-alpha-
contracts/Vault/Swaps.sol#250)
Swaps.quickBurn(address,address,uint256,uint256,uint256) (contracts/mosaic-alpha-contracts/Vault/Swaps.sol#257-289) ignores return value by
IPancakeRouter02(pancakeRouterAddress).swapExactTokensForTokens(amountIn,amountOutMin,path,address(this),deadline) (contracts/mosaic-alpha-
contracts/Vault/Swaps.sol#279)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
```

write-after-write

```
SimpleDeposit._affiliateContract (contracts/mosaic-alpha-contracts/Deposit/SimpleDeposit.sol#35) is written in both
    _affiliateContract = slots[2] (contracts/mosaic-alpha-contracts/Deposit/SimpleDeposit.sol#69)
    _affiliateContract = slots[2] (contracts/mosaic-alpha-contracts/Deposit/SimpleDeposit.sol#72)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#write-after-write
```

Low impact

local-variable-shadowing

```
IPool.poolType().poolType (contracts/mosaic-alpha-contracts/Interfaces/IPool.sol#62) shadows:
    - IPool.poolType() (contracts/mosaic-alpha-contracts/Interfaces/IPool.sol#62) (function)
IPool.vaultAddress().vaultAddress (contracts/mosaic-alpha-contracts/Interfaces/IPool.sol#64) shadows:
    - IPool.vaultAddress() (contracts/mosaic-alpha-contracts/Interfaces/IPool.sol#64) (function)
IPool.isUnlocked().isUnlocked (contracts/mosaic-alpha-contracts/Interfaces/IPool.sol#66) shadows:
    - IPool.isUnlocked() (contracts/mosaic-alpha-contracts/Interfaces/IPool.sol#66) (function)
IPool.poolId().poolId (contracts/mosaic-alpha-contracts/Interfaces/IPool.sol#69) shadows:
    - IPool.poolId() (contracts/mosaic-alpha-contracts/Interfaces/IPool.sol#69) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing

TestToken.constructor(string,string)._name (contracts/test/TestToken.sol#9) shadows:
    - ERC20._name (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#42) (state variable)
TestToken.constructor(string,string)._symbol (contracts/test/TestToken.sol#9) shadows:
    - ERC20._symbol (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#43) (state variable)
```

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing

missing-events-arithmetic

SimpleDeposit.setLockTime(uint64) (contracts/mosaic-alpha-contracts/Deposit/SimpleDeposit.sol#124-126) should emit an event for:
    - lockTime = _time (contracts/mosaic-alpha-contracts/Deposit/SimpleDeposit.sol#125)
SimpleAffiliateBoosterController.setTicketPrice(uint256) (contracts/mosaic-alpha-contracts/User/SimpleAffiliateBoosterController.sol#24-26) should emit an event for:
    - ticketPrice = _price (contracts/mosaic-alpha-contracts/User/SimpleAffiliateBoosterController.sol#25)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-arithmetic

missing-zero-address-validation

PancakeFactory.constructor(address)._feeToSetter (contracts/mosaic-alpha-contracts/helpers/pancake/PancakeFactory.sol#21) lacks a zero-check on :
        - feeToSetter = _feeToSetter (contracts/mosaic-alpha-contracts/helpers/pancake/PancakeFactory.sol#22)
PancakeFactory.setFeeTo(address)._feeTo (contracts/mosaic-alpha-contracts/helpers/pancake/PancakeFactory.sol#46) lacks a zero-check on :
        - feeTo = _feeTo (contracts/mosaic-alpha-contracts/helpers/pancake/PancakeFactory.sol#48)
PancakeFactory.setFeeToSetter(address)._feeToSetter (contracts/mosaic-alpha-contracts/helpers/pancake/PancakeFactory.sol#51) lacks a zero-check on :
        - feeToSetter = _feeToSetter (contracts/mosaic-alpha-contracts/helpers/pancake/PancakeFactory.sol#53)
PancakePair.initialize(address,address)._token0 (contracts/mosaic-alpha-contracts/helpers/pancake/PancakePair.sol#68) lacks a zero-check on :
        - token0 = _token0 (contracts/mosaic-alpha-contracts/helpers/pancake/PancakePair.sol#70)
PancakePair.initialize(address,address)._token1 (contracts/mosaic-alpha-contracts/helpers/pancake/PancakePair.sol#68) lacks a zero-check on :
        - token1 = _token1 (contracts/mosaic-alpha-contracts/helpers/pancake/PancakePair.sol#71)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

PancakeRouter.constructor(address,address)._factory (contracts/mosaic-alpha-contracts/helpers/pancake/PancakeRouter.sol#25) lacks a zero-check on :
        - factory = _factory (contracts/mosaic-alpha-contracts/helpers/pancake/PancakeRouter.sol#26)
PancakeRouter.constructor(address,address)._WETH (contracts/mosaic-alpha-contracts/helpers/pancake/PancakeRouter.sol#25) lacks a zero-check on :
        - WETH = _WETH (contracts/mosaic-alpha-contracts/helpers/pancake/PancakeRouter.sol#27)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

Pool.initialize(string,string,uint8,address,address,address[],uint32[],uint256,IFees.MosaicPoolFees,address,bool)._vaultAddress (contracts/mosaic-alpha-contracts/Pool/Pool.sol#175) lacks a zero-check on :
        - vaultAddress = _vaultAddress (contracts/mosaic-alpha-contracts/Pool/Pool.sol#212)
Pool.initialize(string,string,uint8,address,address,address[],uint32[],uint256,IFees.MosaicPoolFees,address,bool)._owner (contracts/mosaic-alpha-contracts/Pool/Pool.sol#182) lacks a zero-check on :
        - creator = _owner (contracts/mosaic-alpha-contracts/Pool/Pool.sol#232)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

Fees.constructor(address,address)._vault (contracts/mosaic-alpha-contracts/Fees/Fees.sol#77) lacks a zero-check on :
        - vault = _vault (contracts/mosaic-alpha-contracts/Fees/Fees.sol#81)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

TradingBot.constructor(address,address,address).operatorAddress (contracts/mosaic-alpha-bot/Interfaces/TradingBot.sol#17) lacks a zero-check on :
        - _operatorAddress = operatorAddress (contracts/mosaic-alpha-bot/Interfaces/TradingBot.sol#18)
TradingBot.SetOperatorAddress(address).operatorAddress (contracts/mosaic-alpha-bot/Interfaces/TradingBot.sol#26) lacks a zero-check on :
        - _operatorAddress = operatorAddress (contracts/mosaic-alpha-bot/Interfaces/TradingBot.sol#28)
SimpleDeposit.transferOwnership(address).newOwner (contracts/mosaic-alpha-contracts/Deposit/SimpleDeposit.sol#83) lacks a zero-check on :
        - _owner = newOwner (contracts/mosaic-alpha-contracts/Deposit/SimpleDeposit.sol#85)
SimpleDeposit.changeFees(address)._fees (contracts/mosaic-alpha-contracts/Deposit/SimpleDeposit.sol#101) lacks a zero-check on :
        - _feesContract = _fees (contracts/mosaic-alpha-contracts/Deposit/SimpleDeposit.sol#102)
SimpleDeposit.changeAffiliate(address)._affiliate (contracts/mosaic-alpha-contracts/Deposit/SimpleDeposit.sol#109) lacks a zero-check on :
        - _affiliateContract = _affiliate (contracts/mosaic-alpha-contracts/Deposit/SimpleDeposit.sol#110)
FeeBurner.constructor(address,address)._token (contracts/mosaic-alpha-contracts/Fees/FeeBurner.sol#18) lacks a zero-check on :
        - feeToken = _token (contracts/mosaic-alpha-contracts/Fees/FeeBurner.sol#19)
FeeBurner.setFeeToken(address)._token (contracts/mosaic-alpha-contracts/Fees/FeeBurner.sol#78) lacks a zero-check on :
        - feeToken = _token (contracts/mosaic-alpha-contracts/Fees/FeeBurner.sol#79)
FeeBurner.setFeeTo(address)._receiver (contracts/mosaic-alpha-contracts/Fees/FeeBurner.sol#86) lacks a zero-check on :
        - feeTo = _receiver (contracts/mosaic-alpha-contracts/Fees/FeeBurner.sol#87)
SimpleMosaicOracle.initialize(address,address,address,address)._stablecoin (contracts/mosaic-alpha-contracts/Oracle/SimpleMosaicOracle.sol#31) lacks a zero-check on :
        - stablecoin = _stablecoin (contracts/mosaic-alpha-contracts/Oracle/SimpleMosaicOracle.sol#33)
SimpleMosaicOracle.initialize(address,address,address,address)._defaultMiddleHop (contracts/mosaic-alpha-contracts/Oracle/SimpleMosaicOracle.sol#31) lacks a zero-check on :
        - defaultMiddleHop = _defaultMiddleHop (contracts/mosaic-alpha-contracts/Oracle/SimpleMosaicOracle.sol#34)
SimpleMosaicOracle.initialize(address,address,address,address)._vault (contracts/mosaic-alpha-contracts/Oracle/SimpleMosaicOracle.sol#31) lacks a zero-check on :
        - vault = _vault (contracts/mosaic-alpha-contracts/Oracle/SimpleMosaicOracle.sol#36)
SimpleMosaicOracle.setDefaultMiddleHop(address)._defaultMiddleHop (contracts/mosaic-alpha-contracts/Oracle/SimpleMosaicOracle.sol#44) lacks a zero-check on :
        - defaultMiddleHop = _defaultMiddleHop (contracts/mosaic-alpha-contracts/Oracle/SimpleMosaicOracle.sol#46)
PoolFactory.constructor(address,address,address)._vaultAddress (contracts/mosaic-alpha-contracts/Pool/PoolFactory.sol#40) lacks a zero-check on :
        - vaultAddress = _vaultAddress (contracts/mosaic-alpha-contracts/Pool/PoolFactory.sol#48)
PoolFactory.constructor(address,address,address)._poolLibAddress (contracts/mosaic-alpha-contracts/Pool/PoolFactory.sol#44) lacks a zero-check on :
        - poolLibAddress = _poolLibAddress (contracts/mosaic-alpha-contracts/Pool/PoolFactory.sol#50)
PoolFactory.setFeeBurner(address)._burner (contracts/mosaic-alpha-contracts/Pool/PoolFactory.sol#277) lacks a zero-check on :
        - feeBurner = _burner (contracts/mosaic-alpha-contracts/Pool/PoolFactory.sol#278)
QueryApi.constructor(address)._vaultAddr (contracts/mosaic-alpha-contracts/Query/QueryApi.sol#14) lacks a zero-check on :
        - vaultAddr = _vaultAddr (contracts/mosaic-alpha-contracts/Query/QueryApi.sol#15)
QueryApi.setVaultaddress(address)._vaultAddr (contracts/mosaic-alpha-contracts/Query/QueryApi.sol#18) lacks a zero-check on :
        - vaultAddr = _vaultAddr (contracts/mosaic-alpha-contracts/Query/QueryApi.sol#19)
Register.constructor(address,address)._vault (contracts/mosaic-alpha-contracts/Register/Register.sol#20) lacks a zero-check on :
        - vaultAddr = _vault (contracts/mosaic-alpha-contracts/Register/Register.sol#22)
Affiliate.constructor(address,address)._vault (contracts/mosaic-alpha-contracts/User/Affiliate.sol#30) lacks a zero-check on :
        - vault = _vault (contracts/mosaic-alpha-contracts/User/Affiliate.sol#32)
Affiliate.setUserProfileContract(address)._profile (contracts/mosaic-alpha-contracts/User/Affiliate.sol#148) lacks a zero-check on :
        - userProfile = _profile (contracts/mosaic-alpha-contracts/User/Affiliate.sol#149)
Affiliate.setUserOracleContract(address)._oracle (contracts/mosaic-alpha-contracts/User/Affiliate.sol#152) lacks a zero-check on :
        - oracle = _oracle (contracts/mosaic-alpha-contracts/User/Affiliate.sol#153)
Affiliate_Storage.change_oracle(address)._new_oracle (contracts/mosaic-alpha-contracts/User/Affiliate_old.sol#110) lacks a zero-check on :
        - oracle = _new_oracle (contracts/mosaic-alpha-contracts/User/Affiliate_old.sol#111)
SimpleAffiliateBoosterController.constructor(address,address,bool,uint256)._affiliateBooster (contracts/mosaic-alpha-contracts/User/SimpleAffiliateBoosterController.sol#16) lacks a zero-check on :
        - affiliateBooster = _affiliateBooster (contracts/mosaic-alpha-contracts/User/SimpleAffiliateBoosterController.sol#18)
SimpleAffiliateBoosterController.constructor(address,address,bool,uint256)._feeToken (contracts/mosaic-alpha-contracts/User/SimpleAffiliateBoosterController.sol#16) lacks a zero-check on :
        - feeToken = _feeToken (contracts/mosaic-alpha-contracts/User/SimpleAffiliateBoosterController.sol#19)
SimpleAffiliateBoosterController.setFeeToken(address)._a (contracts/mosaic-alpha-contracts/User/SimpleAffiliateBoosterController.sol#28) lacks a zero-check on :
        - feeToken = _a (contracts/mosaic-alpha-contracts/User/SimpleAffiliateBoosterController.sol#29)
SimpleAffiliateBoosterController.setFeeTokenTo(address)._a (contracts/mosaic-alpha-contracts/User/SimpleAffiliateBoosterController.sol#32) lacks a zero-check on :
        - feeTokenTo = _a (contracts/mosaic-alpha-contracts/User/SimpleAffiliateBoosterController.sol#33)
UserProfile.constructor(address,address)._vault (contracts/mosaic-alpha-contracts/User/UserProfile.sol#24) lacks a zero-check on :
        - vault = _vault (contracts/mosaic-alpha-contracts/User/UserProfile.sol#25)
UserProfile.setAffiliateBooster(address)._a (contracts/mosaic-alpha-contracts/User/UserProfile.sol#100) lacks a zero-check on :
        - affiliateBoosterContract = _a (contracts/mosaic-alpha-contracts/User/UserProfile.sol#102)
UserProfile.setFees(address)._a (contracts/mosaic-alpha-contracts/User/UserProfile.sol#106) lacks a zero-check on :
        - fees = _a (contracts/mosaic-alpha-contracts/User/UserProfile.sol#107)
UserProfile.setAffiliate(address)._a (contracts/mosaic-alpha-contracts/User/UserProfile.sol#110) lacks a zero-check on :
        - affiliateContract = _a (contracts/mosaic-alpha-contracts/User/UserProfile.sol#112)
Swaps.constructor(address,address,address,address)._pancakeRouter (contracts/mosaic-alpha-contracts/Vault/Swaps.sol#51) lacks a zero-check on :
        - pancakeRouterAddress = _pancakeRouter (contracts/mosaic-alpha-contracts/Vault/Swaps.sol#52)
Swaps.constructor(address,address,address,address)._vaultAddress (contracts/mosaic-alpha-contracts/Vault/Swaps.sol#51) lacks a zero-check on :

```
        - vaultAddress = _vaultAddress (contracts/mosaic-alpha-contracts/Vault/Swaps.sol#53)
Swaps.constructor(address,address,address,address)._usdtAddress (contracts/mosaic-alpha-contracts/Vault/Swaps.sol#51) lacks a zero-check on :
        - usdtAddress = _usdtAddress (contracts/mosaic-alpha-contracts/Vault/Swaps.sol#54)
SystemWhitelist.transferOwnership(address).newOwner (contracts/mosaic-alpha-contracts/helpers/SystemWhitelist.sol#39) lacks a zero-check on :
        - _owner = newOwner (contracts/mosaic-alpha-contracts/helpers/SystemWhitelist.sol#40)
GovernedContractsMM.constructor(address)._governAddress (contracts/test/ConfigGovernedExample.sol#20) lacks a zero-check on :
        - governAddress = _governAddress (contracts/test/ConfigGovernedExample.sol#21)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
```

incorrect-modifier

```
Modifier GovernedContractsMM.onlyManagers() (contracts/test/ConfigGovernedExample.sol#34-40) does not always execute _; or revert
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-modifier
```

calls-inside-a-loop

```
Pool.updateWeights(uint32,uint32[]) (contracts/mosaic-alpha-contracts/Pool/Pool.sol#667-683) has external calls inside a loop: !
IVault(vaultAddress).isTokenWhitelisted(tokens[i]) (contracts/mosaic-alpha-contracts/Pool/Pool.sol#677)
Pool.getReserves() (contracts/mosaic-alpha-contracts/Pool/Pool.sol#745-752) has external calls inside a loop: _reserves[i] =
IVault(vaultAddress).getInternalBalance(address(this),tokens[i]) (contracts/mosaic-alpha-contracts/Pool/Pool.sol#749)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-inside-a-loop

Governance._execute_Manage(address) (contracts/mosaic-alpha-contracts/Governance/Governance.sol#482-485) has external calls inside a loop:
IGoverned(_contractA).selfManageMe() (contracts/mosaic-alpha-contracts/Governance/Governance.sol#484)
Governance._execute_ManageBytes(address,string,bytes) (contracts/mosaic-alpha-contracts/Governance/Governance.sol#507-524) has external calls inside a
loop: (success,retData) = _contractA.call(abi.encodePacked(signature,_data)) (contracts/mosaic-alpha-contracts/Governance/Governance.sol#522)
Address.functionCallWithValue(address,bytes,uint256,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#128-139) has external calls inside a
loop: (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#137)
Vault._calculateTrailingAndPerformanceFee(address,bool,bool) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#544-572) has external calls inside a loop:
trailingFeesToMint = (IPool(_pool).getTrailingFeeAmount(poolStates[_pool].totalSupplyBase)).toUint128() (contracts/mosaic-alpha-
contracts/Vault/Vault.sol#559)
Vault._consultPoolPricePerLpByAddr(address,IVault.TotalSupplyBase) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#699-704) has external calls inside a
loop: IMosaicOracle(mosaicOracle).consultUsdPoolPricePerLp(poolAddr,ts) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#703)
Vault._calculateTrailingAndPerformanceFee(address,bool,bool) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#544-572) has external calls inside a loop:
performanceFeesToMint = (IPool(_pool).updatePerfFeePricePerLp(tsTmp,valuePerLp)) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#568)
Vault._onBeforeBalanceTransfer(address,address,address,uint256,bool,bool,bool) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#583-615) has external
calls inside a loop: trader = IPool(_pool).creator() (contracts/mosaic-alpha-contracts/Vault/Vault.sol#596)
Vault._onBeforeBalanceTransfer(address,address,address,uint256,bool,bool,bool) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#583-615) has external
calls inside a loop: IFees(fees).onBeforeTransfer(_pool,_from,_to,_internalBalance[_from][_pool],_internalBalance[_to]
[_pool],amount,originalTotalsupplyBase,trader,payload) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#598-601)
Vault._onBeforeBalanceTransfer(address,address,address,uint256,bool,bool,bool) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#583-615) has external
calls inside a loop: IPool(_pool).emitTransfer(address(0),fees,amount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#610)
Vault._onBeforeBalanceTransfer(address,address,address,uint256,bool,bool,bool) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#583-615) has external
calls inside a loop: IFees(fees).onBeforeTransfer(_pool,_from,_to,_internalBalance[_from][_pool],_internalBalance[_to][_pool],amount,0,address(0),payload)
(contracts/mosaic-alpha-contracts/Vault/Vault.sol#614)
Vault._transferInternalBalance(address,address,address,uint256,bool) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#485-506) has external calls inside a
loop: IPool(token).emitTransfer(from,to,amount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#499)
Vault.swap(address,bool,address,address,uint256,bool,uint256,uint64) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#969-1032) has external calls inside
a loop: (_amountOut,_fees) = IPool(poolAddress).queryExactTokensForTokens(tokenIn,tokenOut,_balanceIn,_balanceOut,_amountIn,0) (contracts/mosaic-alpha-
contracts/Vault/Vault.sol#990)
Vault.swap(address,bool,address,address,uint256,bool,uint256,uint64) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#969-1032) has external calls inside
a loop: (_amountOut,_fees) = IPool(poolAddress).queryExactTokensForTokens(tokenIn,tokenOut,_balanceIn,_balanceOut,_amountIn,vaultState.swapFee)
(contracts/mosaic-alpha-contracts/Vault/Vault.sol#990)
Vault.swap(address,bool,address,address,uint256,bool,uint256,uint64) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#969-1032) has external calls inside
a loop: (_amountIn,_fees) = IPool(poolAddress).queryTokensForExactTokens(tokenIn,tokenOut,_balanceIn_scope_0,_balanceOut_scope_1,_amountOut,0)
(contracts/mosaic-alpha-contracts/Vault/Vault.sol#1001)
Vault.swap(address,bool,address,address,uint256,bool,uint256,uint64) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#969-1032) has external calls inside
a loop: (_amountIn,_fees) =
IPool(poolAddress).queryTokensForExactTokens(tokenIn,tokenOut,_balanceIn_scope_0,_balanceOut_scope_1,_amountOut,vaultState.swapFee) (contracts/mosaic-
alpha-contracts/Vault/Vault.sol#1001)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-inside-a-loop

SimpleMosaicOracle._getPairAddress(address,address) (contracts/mosaic-alpha-contracts/Oracle/SimpleMosaicOracle.sol#238-246) has external calls inside a
loop: pair = IPancakeFactory(pancakeRouter.factory()).getPair(tokenA,tokenB) (contracts/mosaic-alpha-contracts/Oracle/SimpleMosaicOracle.sol#240)
SimpleMosaicOracle._getDirectPairPrice(address,address,uint256) (contracts/mosaic-alpha-contracts/Oracle/SimpleMosaicOracle.sol#210-226) has external calls
inside a loop: (reserveA,reserveB) = pairContract.getReserves() (contracts/mosaic-alpha-contracts/Oracle/SimpleMosaicOracle.sol#219)
SimpleMosaicOracle._getDirectPairPrice(address,address,uint256) (contracts/mosaic-alpha-contracts/Oracle/SimpleMosaicOracle.sol#210-226) has external calls
inside a loop: tokenA == pairContract.token0() (contracts/mosaic-alpha-contracts/Oracle/SimpleMosaicOracle.sol#221)
SimpleMosaicOracle._consultPairAddress(address,address) (contracts/mosaic-alpha-contracts/Oracle/SimpleMosaicOracle.sol#385-393) has external calls inside
a loop: pair = IPancakeFactory(pancakeRouter.factory()).getPair(tokenA,tokenB) (contracts/mosaic-alpha-contracts/Oracle/SimpleMosaicOracle.sol#387)
SimpleMosaicOracle._consultDirectPairPrice(address,address,uint256) (contracts/mosaic-alpha-contracts/Oracle/SimpleMosaicOracle.sol#367-383) has external
calls inside a loop: (reserveA,reserveB) = pairContract.getReserves() (contracts/mosaic-alpha-contracts/Oracle/SimpleMosaicOracle.sol#376)
SimpleMosaicOracle._consultDirectPairPrice(address,address,uint256) (contracts/mosaic-alpha-contracts/Oracle/SimpleMosaicOracle.sol#367-383) has external
calls inside a loop: tokenA == pairContract.token0() (contracts/mosaic-alpha-contracts/Oracle/SimpleMosaicOracle.sol#378)
PoolFactory.create(string,string,uint8,address[],uint32[],uint256[],IFees.MosaicPoolFees,address,bool) (contracts/mosaic-alpha-
contracts/Pool/PoolFactory.sol#120-263) has external calls inside a loop: IERC20(tokens[i]).transferFrom(msg.sender,address(this),initialLiquidity[i])
(contracts/mosaic-alpha-contracts/Pool/PoolFactory.sol#196-200)
PoolFactory.create(string,string,uint8,address[],uint32[],uint256[],IFees.MosaicPoolFees,address,bool) (contracts/mosaic-alpha-
contracts/Pool/PoolFactory.sol#120-263) has external calls inside a loop: IERC20(tokens[i]).approve(vaultAddress,initialLiquidity[i]) (contracts/mosaic-
alpha-contracts/Pool/PoolFactory.sol#203)
PoolFactory.create(string,string,uint8,address[],uint32[],uint256[],IFees.MosaicPoolFees,address,bool) (contracts/mosaic-alpha-
contracts/Pool/PoolFactory.sol#120-263) has external calls inside a loop: IVault(vaultAddress).depositToInternalBalance(tokens[i],initialLiquidity[i])
(contracts/mosaic-alpha-contracts/Pool/PoolFactory.sol#206-209)
PoolFactory.create(string,string,uint8,address[],uint32[],uint256[],IFees.MosaicPoolFees,address,bool) (contracts/mosaic-alpha-
contracts/Pool/PoolFactory.sol#120-263) has external calls inside a loop:
IVault(vaultAddress).transferInternalBalance(tokens[i],poolAddr,initialLiquidity[i]) (contracts/mosaic-alpha-contracts/Pool/PoolFactory.sol#212-216)
QueryApi.getPool(uint32) (contracts/mosaic-alpha-contracts/Query/QueryApi.sol#23-43) has external calls inside a loop: require(bool,string)(poolId > 0 &&
poolId <= IVault(vaultAddr).poolCount(),INVALID POOL ID) (contracts/mosaic-alpha-contracts/Query/QueryApi.sol#24)
QueryApi.getPool(uint32) (contracts/mosaic-alpha-contracts/Query/QueryApi.sol#23-43) has external calls inside a loop: poolAddr =
IVault(vaultAddr).poolIdToAddress(poolId) (contracts/mosaic-alpha-contracts/Query/QueryApi.sol#25)
QueryApi.getPool(uint32) (contracts/mosaic-alpha-contracts/Query/QueryApi.sol#23-43) has external calls inside a loop: _poolInfo.name =
IPool(poolAddr).name() (contracts/mosaic-alpha-contracts/Query/QueryApi.sol#28)
QueryApi.getPool(uint32) (contracts/mosaic-alpha-contracts/Query/QueryApi.sol#23-43) has external calls inside a loop: _poolInfo.owner =
IPool(poolAddr).owner() (contracts/mosaic-alpha-contracts/Query/QueryApi.sol#29)
QueryApi.getPool(uint32) (contracts/mosaic-alpha-contracts/Query/QueryApi.sol#23-43) has external calls inside a loop: _poolInfo.poolType =
IPool(poolAddr).poolType() (contracts/mosaic-alpha-contracts/Query/QueryApi.sol#31)
QueryApi.getPool(uint32) (contracts/mosaic-alpha-contracts/Query/QueryApi.sol#23-43) has external calls inside a loop: tmp = IPool(poolAddr).getPoolFees()
(contracts/mosaic-alpha-contracts/Query/QueryApi.sol#33)
QueryApi.getPool(uint32) (contracts/mosaic-alpha-contracts/Query/QueryApi.sol#23-43) has external calls inside a loop: _poolInfo.swapFee =
IVault(vaultAddr).getVaultState().swapFee (contracts/mosaic-alpha-contracts/Query/QueryApi.sol#34)
QueryApi.getPool(uint32) (contracts/mosaic-alpha-contracts/Query/QueryApi.sol#23-43) has external calls inside a loop: _poolInfo.unlocked =
IPool(poolAddr).isUnlocked() (contracts/mosaic-alpha-contracts/Query/QueryApi.sol#38)
QueryApi.getPool(uint32) (contracts/mosaic-alpha-contracts/Query/QueryApi.sol#23-43) has external calls inside a loop: _poolInfo.tokens =
IPool(poolAddr).getTokens() (contracts/mosaic-alpha-contracts/Query/QueryApi.sol#39)
QueryApi.getPool(uint32) (contracts/mosaic-alpha-contracts/Query/QueryApi.sol#23-43) has external calls inside a loop: _poolInfo.weights =
IPool(poolAddr).getWeights() (contracts/mosaic-alpha-contracts/Query/QueryApi.sol#40)
QueryApi.getPool(uint32) (contracts/mosaic-alpha-contracts/Query/QueryApi.sol#23-43) has external calls inside a loop: _poolInfo.reserves =
IPool(poolAddr).getReserves() (contracts/mosaic-alpha-contracts/Query/QueryApi.sol#41)
Register._approvePool(address,bool) (contracts/mosaic-alpha-contracts/Register/Register.sol#57-67) has external calls inside a loop: poolId =
IVault(vaultAddr).poolAddressToId(_poolAddr) (contracts/mosaic-alpha-contracts/Register/Register.sol#58)
Register.approvePoolsById(uint32[]) (contracts/mosaic-alpha-contracts/Register/Register.sol#39-43) has external calls inside a loop:
_approvePool(IVault(vaultAddr).poolIdToAddress(_poolIds[i]),true) (contracts/mosaic-alpha-contracts/Register/Register.sol#41)
Register.revokePoolsById(uint32[]) (contracts/mosaic-alpha-contracts/Register/Register.sol#51-55) has external calls inside a loop:
_approvePool(IVault(vaultAddr).poolIdToAddress(_poolIds[i]),false) (contracts/mosaic-alpha-contracts/Register/Register.sol#53)
```

```
Affiliate.userStakeChanged(address,address,uint256) (contracts/mosaic-alpha-contracts/User/Affiliate.sol#163-191) has external calls inside a loop:
IUserProfile(userProfile).setUserRank(user,rank - 1) (contracts/mosaic-alpha-contracts/User/Affiliate.sol#176)
Swaps._approveToken(address,uint256) (contracts/mosaic-alpha-contracts/Vault/Swaps.sol#24-27) has external calls inside a loop:
IERC20(_token).approve(vaultAddress,_amount) (contracts/mosaic-alpha-contracts/Vault/Swaps.sol#25)
Swaps._approveToken(address,uint256) (contracts/mosaic-alpha-contracts/Vault/Swaps.sol#24-27) has external calls inside a loop:
IERC20(_token).approve(pancakeRouterAddress,_amount) (contracts/mosaic-alpha-contracts/Vault/Swaps.sol#26)
Swaps.quickQuoteMint(address,address,uint256) (contracts/mosaic-alpha-contracts/Vault/Swaps.sol#70-88) has external calls inside a loop: _usdNeeded +=
IPancakeRouter02(pancakeRouterAddress).getAmountsIn(_requestedAmounts[i],path)[0] (contracts/mosaic-alpha-contracts/Vault/Swaps.sol#80)
Swaps.quickQuoteBurn(address,address,uint256) (contracts/mosaic-alpha-contracts/Vault/Swaps.sol#91-108) has external calls inside a loop: usdToReceive +=
IPancakeRouter02(pancakeRouterAddress).getAmountsOut(_amounts[i],path)[1] (contracts/mosaic-alpha-contracts/Vault/Swaps.sol#100)
Swaps.quoteMint(address,address[][],uint256) (contracts/mosaic-alpha-contracts/Vault/Swaps.sol#110-125) has external calls inside a loop: _usdNeeded +=
IPancakeRouter02(pancakeRouterAddress).getAmountsIn(_requestedAmounts[i],_routing[i])[0] (contracts/mosaic-alpha-contracts/Vault/Swaps.sol#117)
Swaps.quoteBurn(address,address[][],uint256) (contracts/mosaic-alpha-contracts/Vault/Swaps.sol#127-141) has external calls inside a loop: _usdReceived +=
IPancakeRouter02(pancakeRouterAddress).getAmountsOut(_amounts[i],_routing[i])[_routing[i].length - 1] (contracts/mosaic-alpha-
contracts/Vault/Swaps.sol#134)
Swaps.mint(address,address,address[][],uint256,uint256,address,uint256) (contracts/mosaic-alpha-contracts/Vault/Swaps.sol#144-189) has external calls
inside a loop: amountInMax = IPancakeRouter02(pancakeRouterAddress).getAmountsIn(amountOut,path)[0] (contracts/mosaic-alpha-contracts/Vault/Swaps.sol#163)
Swaps.mint(address,address,address[][],uint256,uint256,address,uint256) (contracts/mosaic-alpha-contracts/Vault/Swaps.sol#144-189) has external calls
inside a loop: IPancakeRouter02(pancakeRouterAddress).swapTokensForExactTokens(amountOut,amountInMax,path,to,deadline) (contracts/mosaic-alpha-
contracts/Vault/Swaps.sol#165)
Swaps.mint(address,address,address[][],uint256,uint256,address,uint256) (contracts/mosaic-alpha-contracts/Vault/Swaps.sol#144-189) has external calls
inside a loop: IVault(vaultAddress).depositToInternalBalance(token,amountOut) (contracts/mosaic-alpha-contracts/Vault/Swaps.sol#168)
Swaps.mint(address,address,address[][],uint256,uint256,address,uint256) (contracts/mosaic-alpha-contracts/Vault/Swaps.sol#144-189) has external calls
inside a loop: IVault(vaultAddress).depositToInternalBalance(usdAddr,_requestedAmounts[i]) (contracts/mosaic-alpha-contracts/Vault/Swaps.sol#174)
Swaps.burn(address,address,address[][],uint256,uint256,uint256) (contracts/mosaic-alpha-contracts/Vault/Swaps.sol#191-220) has external calls inside a
loop: amountOutMin = IPancakeRouter02(pancakeRouterAddress).getAmountsOut(amountsToReceive[i],path)[path.length - 1] (contracts/mosaic-alpha-
contracts/Vault/Swaps.sol#209)
Swaps.burn(address,address,address[][],uint256,uint256,uint256) (contracts/mosaic-alpha-contracts/Vault/Swaps.sol#191-220) has external calls inside a
loop: IPancakeRouter02(pancakeRouterAddress).swapExactTokensForTokens(amountIn,amountOutMin,path,address(this),deadline) (contracts/mosaic-alpha-
contracts/Vault/Swaps.sol#210)
Swaps.quickMint(address,address,uint256,uint256,address,uint256) (contracts/mosaic-alpha-contracts/Vault/Swaps.sol#222-255) has external calls inside a
loop: amountInMax = IPancakeRouter02(pancakeRouterAddress).getAmountsIn(_requestedAmounts[i],path)[0] (contracts/mosaic-alpha-
contracts/Vault/Swaps.sol#238)
Swaps.quickMint(address,address,uint256,uint256,address,uint256) (contracts/mosaic-alpha-contracts/Vault/Swaps.sol#222-255) has external calls inside a
loop: require(bool,string)(amountInMax <= IERC20(usdAddr).balanceOf(address(this)),MAX SLIPPAGE EXCEEDED) (contracts/mosaic-alpha-
contracts/Vault/Swaps.sol#239)
Swaps.quickMint(address,address,uint256,uint256,address,uint256) (contracts/mosaic-alpha-contracts/Vault/Swaps.sol#222-255) has external calls inside a
loop: IPancakeRouter02(pancakeRouterAddress).swapTokensForExactTokens(amountOut,amountInMax,path,swapAddr,deadline) (contracts/mosaic-alpha-
contracts/Vault/Swaps.sol#240)
Swaps.quickMint(address,address,uint256,uint256,address,uint256) (contracts/mosaic-alpha-contracts/Vault/Swaps.sol#222-255) has external calls inside a
loop: IVault(vaultAddress).depositToInternalBalance(_tokens[i],amountOut) (contracts/mosaic-alpha-contracts/Vault/Swaps.sol#243)
Swaps.quickBurn(address,address,uint256,uint256,uint256) (contracts/mosaic-alpha-contracts/Vault/Swaps.sol#257-289) has external calls inside a loop:
amountOutMin = IPancakeRouter02(pancakeRouterAddress).getAmountsOut(amountsToReceive[i],path)[1] (contracts/mosaic-alpha-contracts/Vault/Swaps.sol#278)
Swaps.quickBurn(address,address,uint256,uint256,uint256) (contracts/mosaic-alpha-contracts/Vault/Swaps.sol#257-289) has external calls inside a loop:
IPancakeRouter02(pancakeRouterAddress).swapExactTokensForTokens(amountIn,amountOutMin,path,address(this),deadline) (contracts/mosaic-alpha-
contracts/Vault/Swaps.sol#279)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-inside-a-loop
```

reentrancy-vulnerabilities-2

```
Reentrancy in PancakePair.burn(address) (contracts/mosaic-alpha-contracts/helpers/pancake/PancakePair.sol#136-158):
    External calls:
    - _safeTransfer(_token0,to,amount0) (contracts/mosaic-alpha-contracts/helpers/pancake/PancakePair.sol#150)
        - (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value)) (contracts/mosaic-alpha-contracts/helpers/pancake/PancakePair.sol#47)
    - _safeTransfer(_token1,to,amount1) (contracts/mosaic-alpha-contracts/helpers/pancake/PancakePair.sol#151)
        - (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value)) (contracts/mosaic-alpha-contracts/helpers/pancake/PancakePair.sol#47)
    State variables written after the call(s):
    - _update(balance0,balance1,_reserve0,_reserve1) (contracts/mosaic-alpha-contracts/helpers/pancake/PancakePair.sol#155)
        - price0CumulativeLast += uint256(UQ112x112.encode(_reserve1).uqdiv(_reserve0)) * timeElapsed (contracts/mosaic-alpha-
contracts/helpers/pancake/PancakePair.sol#81)
    - _update(balance0,balance1,_reserve0,_reserve1) (contracts/mosaic-alpha-contracts/helpers/pancake/PancakePair.sol#155)
        - price1CumulativeLast += uint256(UQ112x112.encode(_reserve0).uqdiv(_reserve1)) * timeElapsed (contracts/mosaic-alpha-
contracts/helpers/pancake/PancakePair.sol#82)
Reentrancy in PancakeFactory.createPair(address,address) (contracts/mosaic-alpha-contracts/helpers/pancake/PancakeFactory.sol#29-44):
    External calls:
    - IPancakePair(pair).initialize(token0,token1) (contracts/mosaic-alpha-contracts/helpers/pancake/PancakeFactory.sol#39)
    State variables written after the call(s):
    - allPairs.push(pair) (contracts/mosaic-alpha-contracts/helpers/pancake/PancakeFactory.sol#42)
Reentrancy in PancakePair.swap(uint256,uint256,address,bytes) (contracts/mosaic-alpha-contracts/helpers/pancake/PancakePair.sol#161-189):
    External calls:
    - _safeTransfer(_token0,to,amount0Out) (contracts/mosaic-alpha-contracts/helpers/pancake/PancakePair.sol#172)
        - (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value)) (contracts/mosaic-alpha-contracts/helpers/pancake/PancakePair.sol#47)
    - _safeTransfer(_token1,to,amount1Out) (contracts/mosaic-alpha-contracts/helpers/pancake/PancakePair.sol#173)
        - (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value)) (contracts/mosaic-alpha-contracts/helpers/pancake/PancakePair.sol#47)
    - IPancakeCallee(to).pancakeCall(msg.sender,amount0Out,amount1Out,data) (contracts/mosaic-alpha-contracts/helpers/pancake/PancakePair.sol#174)
    State variables written after the call(s):
    - _update(balance0,balance1,_reserve0,_reserve1) (contracts/mosaic-alpha-contracts/helpers/pancake/PancakePair.sol#187)
        - price0CumulativeLast += uint256(UQ112x112.encode(_reserve1).uqdiv(_reserve0)) * timeElapsed (contracts/mosaic-alpha-
contracts/helpers/pancake/PancakePair.sol#81)
    - _update(balance0,balance1,_reserve0,_reserve1) (contracts/mosaic-alpha-contracts/helpers/pancake/PancakePair.sol#187)
        - price1CumulativeLast += uint256(UQ112x112.encode(_reserve0).uqdiv(_reserve1)) * timeElapsed (contracts/mosaic-alpha-
contracts/helpers/pancake/PancakePair.sol#82)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2

Reentrancy in Pool.initialize(string,string,uint8,address,address,address[],uint32[],uint256,IFees.MosaicPoolFees,address,bool) (contracts/mosaic-alpha-
contracts/Pool/Pool.sol#171-235):
    External calls:
    - IVault(_vaultAddress).disableFees() (contracts/mosaic-alpha-contracts/Pool/Pool.sol#202)
    State variables written after the call(s):
    - decimals = 18 (contracts/mosaic-alpha-contracts/Pool/Pool.sol#210)
    - feeless = _feeless (contracts/mosaic-alpha-contracts/Pool/Pool.sol#204)
    - feesAddress = _feesAddress (contracts/mosaic-alpha-contracts/Pool/Pool.sol#213)
    - initialSupply = _lpTokens (contracts/mosaic-alpha-contracts/Pool/Pool.sol#219)
    - isUnlocked = true (contracts/mosaic-alpha-contracts/Pool/Pool.sol#217)
    - name = string(abi.encodePacked(MosaicAlpha: ,_name)) (contracts/mosaic-alpha-contracts/Pool/Pool.sol#205)
    - newWeights = _weights (contracts/mosaic-alpha-contracts/Pool/Pool.sol#216)
    - poolId = IVault(_vaultAddress).poolAddressToId(address(this)) (contracts/mosaic-alpha-contracts/Pool/Pool.sol#206)
    - poolType = _poolType (contracts/mosaic-alpha-contracts/Pool/Pool.sol#211)
    - symbol = _symbol (contracts/mosaic-alpha-contracts/Pool/Pool.sol#209)
    - tokens = _tokens (contracts/mosaic-alpha-contracts/Pool/Pool.sol#214)
    - vaultAddress = _vaultAddress (contracts/mosaic-alpha-contracts/Pool/Pool.sol#212)
    - weights = _weights (contracts/mosaic-alpha-contracts/Pool/Pool.sol#215)
Reentrancy in Pool.initialize(string,string,uint8,address,address,address[],uint32[],uint256,IFees.MosaicPoolFees,address,bool) (contracts/mosaic-alpha-
contracts/Pool/Pool.sol#171-235):
    External calls:
    - IVault(_vaultAddress).disableFees() (contracts/mosaic-alpha-contracts/Pool/Pool.sol#202)
    - IVault(vaultAddress).registerTokens(tokens,true) (contracts/mosaic-alpha-contracts/Pool/Pool.sol#224)
    State variables written after the call(s):
    - _transferOwnership(_owner) (contracts/mosaic-alpha-contracts/Pool/Pool.sol#234)
        - _ownr = newOwner (contracts/mosaic-alpha-contracts/Pool/Pool.sol#152)
    - creator = _owner (contracts/mosaic-alpha-contracts/Pool/Pool.sol#232)
Reentrancy in Pool.removeToken(uint256) (contracts/mosaic-alpha-contracts/Pool/Pool.sol#569-590):
    External calls:
    - IVault(vaultAddress).deregisterToken(tokens[_index],_remainingBalance) (contracts/mosaic-alpha-contracts/Pool/Pool.sol#581)
    State variables written after the call(s):
    - weights[_index] = weights[weights.length - 1] (contracts/mosaic-alpha-contracts/Pool/Pool.sol#586)
```

```
    - weights.pop() (contracts/mosaic-alpha-contracts/Pool/Pool.sol#587)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2

Reentrancy in Vault._depositToInternalBalance(address,address,address,uint256,bool) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#459-466):
    External calls:
    - IERC20(token).safeTransferFrom(from,address(this),amount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#463)
    State variables written after the call(s):
    - _internalBalance[to][token] += amount (contracts/mosaic-alpha-contracts/Vault/Vault.sol#464)
Reentrancy in Vault._onBeforeBalanceTransfer(address,address,address,uint256,bool,bool,bool) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#583-615):
    External calls:
    - (payload.trailingLpToMint,payload.performanceLpToMint) =
_calculateTrailingAndPerformanceFee(_pool,forceTrailingFeeAllocation,forcePerformanceFeeAllocation) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#591)
        - IMosaicOracle(mosaicOracle).consultUsdPoolPricePerLp(poolAddr,ts) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#703)
        - performanceFeesToMint = (IPool(_pool).updatePerfFeePricePerLp(tsTmp,valuePerLp)) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#568)
    - IFees(fees).onBeforeTransfer(_pool,_from,_to,_internalBalance[_from][_pool],_internalBalance[_to]
[_pool],amount,originalTotalsupplyBase,trader,payload) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#598-601)
    State variables written after the call(s):
    - _handlerForPoolOfUser(fees,_pool,true) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#607)
        - poolOfUser[user].push(poolId) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1226)
        - poolOfUser[user][index_scope_0] = replacementPoolId (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1240)
        - poolOfUser[user].pop() (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1241)
    - _handlerForPoolOfUser(fees,_pool,true) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#607)
        - poolOfUserIndexes[user][poolId] = (poolOfUser[user].length).toUint32() (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1228)
        - poolOfUserIndexes[user][replacementPoolId] = uint32(index_scope_0) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1239)
        - poolOfUserIndexes[user][poolId] = 0 (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1242)
Reentrancy in Vault.registerPool(address,address,address) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#721-739):
    External calls:
    - _registerUser(_user,_referredBy) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#727)
        - IUserProfile(userProfile).registerUser(to,referredBy) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#202)
    State variables written after the call(s):
    - _poolIdToAddress[_poolId] = _pool (contracts/mosaic-alpha-contracts/Vault/Vault.sol#733)
    - poolCount += 1 (contracts/mosaic-alpha-contracts/Vault/Vault.sol#729)
    - usrOwnedPools[_user].push(_poolId) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#735)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2

Reentrancy in Fees._selfManageMeBefore() (contracts/mosaic-alpha-contracts/Fees/Fees.sol#145-161):
    External calls:
    - IUserProfile(userProfile).registerUser(slots[1]) (contracts/mosaic-alpha-contracts/Fees/Fees.sol#148)
    State variables written after the call(s):
    - _transferOwnership(slots[0]) (contracts/mosaic-alpha-contracts/Fees/Fees.sol#152)
        - _owner = newOwner (contracts/mosaic-alpha-contracts/helpers/Ownable.sol#78)
    - affiliate = slots[2] (contracts/mosaic-alpha-contracts/Fees/Fees.sol#149)
    - claimLimitAdapter = slots[1] (contracts/mosaic-alpha-contracts/Fees/Fees.sol#159)
    - claimLimitAdapter = address(0) (contracts/mosaic-alpha-contracts/Fees/Fees.sol#160)
    - deposit = slots[9] (contracts/mosaic-alpha-contracts/Fees/Fees.sol#151)
    - feeTo = slots[6] (contracts/mosaic-alpha-contracts/Fees/Fees.sol#150)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
```

reentrancy-vulnerabilities-3

```
Reentrancy in AffiliateBooster._allocateTicket(address,address,uint256,uint256) (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#352-365):
    External calls:
    - userProfile.registerUser(_a,_referredBy) (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#353)
    State variables written after the call(s):
    - epochTickets[state.currentEpoch].push(_a) (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#355)
    - epochUserTickets[state.currentEpoch][_a] += uint64(_c) (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#359)
    - epochs[state.currentEpoch].ticketsBought += uint64(_c) (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#358)
    - state.totalPlayerWeight += state.epochWeights[0] (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#356)
    - state.totalTicketCount += uint64(_c) (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#360)
Reentrancy in SimpleDeposit._deposit(address,address,uint256) (contracts/mosaic-alpha-contracts/Deposit/SimpleDeposit.sol#269-293):
    External calls:
    - successTransfer = _stakingToken.transferFrom(_sender,address(this),_amount) (contracts/mosaic-alpha-contracts/Deposit/SimpleDeposit.sol#279-283)
    State variables written after the call(s):
    - accountBalances[_user].balance = accountBalances[_user].balance + _amount (contracts/mosaic-alpha-contracts/Deposit/SimpleDeposit.sol#289)
    - totalDeposits = totalDeposits + _amount (contracts/mosaic-alpha-contracts/Deposit/SimpleDeposit.sol#290)
Reentrancy in Affiliate._registerUserPurchase(address,address,address,uint256) (contracts/mosaic-alpha-contracts/User/Affiliate.sol#220-279):
    External calls:
    - IUserProfile(userProfile).registerUser(user,referredBy) (contracts/mosaic-alpha-contracts/User/Affiliate.sol#221)
    - IUserProfile(userProfile).registerUser(trader,address(0)) (contracts/mosaic-alpha-contracts/User/Affiliate.sol#223)
    State variables written after the call(s):
    - userData[user].userPurchase += usdAmount (contracts/mosaic-alpha-contracts/User/Affiliate.sol#226)
    - userData[referredBy].referralPurchase += usdAmount (contracts/mosaic-alpha-contracts/User/Affiliate.sol#229)
    - userData[referredBy].activeReferralCount ++ (contracts/mosaic-alpha-contracts/User/Affiliate.sol#231)
Reentrancy in AffiliateBooster.pickNextReferral() (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#372-410):
    External calls:
    - closeEpochIfTimeUp() (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#372)
        - controller.epochClosed(state.currentEpoch - 1) (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#427)
    State variables written after the call(s):
    - epochUserTickets[state.currentEpoch - i][pickedAddress] -= 1 (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#393)
Reentrancy in Affiliate.userStakeChanged(address,address,uint256) (contracts/mosaic-alpha-contracts/User/Affiliate.sol#163-191):
    External calls:
    - IUserProfile(userProfile).registerUser(user,referredBy) (contracts/mosaic-alpha-contracts/User/Affiliate.sol#164)
    State variables written after the call(s):
    - userData[user].kdxStake = kdxAmount (contracts/mosaic-alpha-contracts/User/Affiliate.sol#166)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2

Reentrancy in PancakePair.burn(address) (contracts/mosaic-alpha-contracts/helpers/pancake/PancakePair.sol#136-158):
    External calls:
    - _safeTransfer(_token0,to,amount0) (contracts/mosaic-alpha-contracts/helpers/pancake/PancakePair.sol#150)
        - (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value)) (contracts/mosaic-alpha-contracts/helpers/pancake/PancakePair.sol#47)
    - _safeTransfer(_token1,to,amount1) (contracts/mosaic-alpha-contracts/helpers/pancake/PancakePair.sol#151)
        - (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value)) (contracts/mosaic-alpha-contracts/helpers/pancake/PancakePair.sol#47)
    Event emitted after the call(s):
    - Burn(msg.sender,amount0,amount1,to) (contracts/mosaic-alpha-contracts/helpers/pancake/PancakePair.sol#157)
    - Sync(reserve0,reserve1) (contracts/mosaic-alpha-contracts/helpers/pancake/PancakePair.sol#87)
        - _update(balance0,balance1,_reserve0,_reserve1) (contracts/mosaic-alpha-contracts/helpers/pancake/PancakePair.sol#155)
Reentrancy in PancakeFactory.createPair(address,address) (contracts/mosaic-alpha-contracts/helpers/pancake/PancakeFactory.sol#29-44):
    External calls:
    - IPancakePair(pair).initialize(token0,token1) (contracts/mosaic-alpha-contracts/helpers/pancake/PancakeFactory.sol#39)
    Event emitted after the call(s):
    - PairCreated(token0,token1,pair,allPairs.length) (contracts/mosaic-alpha-contracts/helpers/pancake/PancakeFactory.sol#43)
Reentrancy in PancakePair.swap(uint256,uint256,address,bytes) (contracts/mosaic-alpha-contracts/helpers/pancake/PancakePair.sol#161-189):
    External calls:
    - _safeTransfer(_token0,to,amount0Out) (contracts/mosaic-alpha-contracts/helpers/pancake/PancakePair.sol#172)
        - (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value)) (contracts/mosaic-alpha-contracts/helpers/pancake/PancakePair.sol#47)
    - _safeTransfer(_token1,to,amount1Out) (contracts/mosaic-alpha-contracts/helpers/pancake/PancakePair.sol#173)
        - (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value)) (contracts/mosaic-alpha-contracts/helpers/pancake/PancakePair.sol#47)
    - IPancakeCallee(to).pancakeCall(msg.sender,amount0Out,amount1Out,data) (contracts/mosaic-alpha-contracts/helpers/pancake/PancakePair.sol#174)
    Event emitted after the call(s):
    - Swap(msg.sender,amount0In,amount1In,amount0Out,amount1Out,to) (contracts/mosaic-alpha-contracts/helpers/pancake/PancakePair.sol#188)
    - Sync(reserve0,reserve1) (contracts/mosaic-alpha-contracts/helpers/pancake/PancakePair.sol#87)
        - _update(balance0,balance1,_reserve0,_reserve1) (contracts/mosaic-alpha-contracts/helpers/pancake/PancakePair.sol#187)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3

Reentrancy in Pool._transfer(address,uint256) (contracts/mosaic-alpha-contracts/Pool/Pool.sol#289-293):
    External calls:
```

```
    - IVault(vaultAddress).transferFromAsTokenContract(msg.sender,_to,_value) (contracts/mosaic-alpha-contracts/Pool/Pool.sol#290)
    Event emitted after the call(s):
    - Transfer(msg.sender,_to,_value) (contracts/mosaic-alpha-contracts/Pool/Pool.sol#291)
Reentrancy in Pool._transferFrom(address,address,uint256) (contracts/mosaic-alpha-contracts/Pool/Pool.sol#304-310):
    External calls:
    - IVault(vaultAddress).transferFromAsTokenContract(_from,_to,_value) (contracts/mosaic-alpha-contracts/Pool/Pool.sol#307)
    Event emitted after the call(s):
    - Transfer(_from,_to,_value) (contracts/mosaic-alpha-contracts/Pool/Pool.sol#308)
Reentrancy in Pool.initialize(string,string,uint8,address,address,address[],uint32[],uint256,IFees.MosaicPoolFees,address,bool) (contracts/mosaic-alpha-
contracts/Pool/Pool.sol#171-235):
    External calls:
    - IVault(_vaultAddress).disableFees() (contracts/mosaic-alpha-contracts/Pool/Pool.sol#202)
    - IVault(vaultAddress).registerTokens(tokens,true) (contracts/mosaic-alpha-contracts/Pool/Pool.sol#224)
    Event emitted after the call(s):
    - OwnershipTransferred(oldOwner,newOwner) (contracts/mosaic-alpha-contracts/Pool/Pool.sol#153)
        - _transferOwnership(_owner) (contracts/mosaic-alpha-contracts/Pool/Pool.sol#234)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3

Reentrancy in Vault.Burn(uint32,uint256,uint256[],bool,uint256,address) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#913-951):
    External calls:
    - (amountsToReceive,trailingFeeLpTokens) = IPool(_poolAddr)._burn(LPTokensToBurn,poolStates[_poolAddr].totalSupplyBase) (contracts/mosaic-alpha-
contracts/Vault/Vault.sol#921)
    - _transferInternalBalance(tokens[i],_poolAddr,msg.sender,amountsToReceive[i]) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#927)
        - IMosaicOracle(mosaicOracle).consultUsdPoolPricePerLp(poolAddr,ts) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#703)
        - IFees(fees).onBeforeTransfer(_pool,_from,_to,_internalBalance[_from][_pool],_internalBalance[_to]
[_pool],amount,originalTotalsupplyBase,trader,payload) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#598-601)
        - IPool(token).emitTransfer(from,to,amount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#499)
        - performanceFeesToMint = (IPool(_pool).updatePerfFeePricePerLp(tsTmp,valuePerLp)) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#568)
        - IPool(_pool).emitTransfer(address(0),fees,amount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#610)
        - IFees(fees).onBeforeTransfer(_pool,_from,_to,_internalBalance[_from][_pool],_internalBalance[_to][_pool],amount,0,address(0),payload)
(contracts/mosaic-alpha-contracts/Vault/Vault.sol#614)
    Event emitted after the call(s):
    - InternalBalanceChanged(from,token,- int256(amount)) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#489)
        - _transferInternalBalance(tokens[i],_poolAddr,msg.sender,amountsToReceive[i]) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#927)
    - InternalBalanceChanged(to,token,int256(amount)) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#490)
        - _transferInternalBalance(tokens[i],_poolAddr,msg.sender,amountsToReceive[i]) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#927)
    - InternalBalanceChanged(fees,_pool,int256(uint256(expAmount))) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#609)
        - _transferInternalBalance(tokens[i],_poolAddr,msg.sender,amountsToReceive[i]) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#927)
    - TotalSupplyBaseChanged(_pool,poolStates[_pool].totalSupplyBase) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#605)
        - _transferInternalBalance(tokens[i],_poolAddr,msg.sender,amountsToReceive[i]) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#927)
Reentrancy in Vault.Burn(uint32,uint256,uint256[],bool,uint256,address) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#913-951):
    External calls:
    - (amountsToReceive,trailingFeeLpTokens) = IPool(_poolAddr)._burn(LPTokensToBurn,poolStates[_poolAddr].totalSupplyBase) (contracts/mosaic-alpha-
contracts/Vault/Vault.sol#921)
    - _withdrawFromInternalBalance(tokens[i],_poolAddr,msg.sender,amountsToReceive[i]) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#932)
        - returndata = address(token).functionCall(data,SafeERC20: low-level call failed)
(node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#110)
        - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#137)
        - IERC20(token).safeTransfer(to,amount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#476)
    External calls sending eth:
    - _withdrawFromInternalBalance(tokens[i],_poolAddr,msg.sender,amountsToReceive[i]) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#932)
        - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#137)
    Event emitted after the call(s):
    - InternalBalanceChanged(from,token,- int256(amount)) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#478)
        - _withdrawFromInternalBalance(tokens[i],_poolAddr,msg.sender,amountsToReceive[i]) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#932)
Reentrancy in Vault.Burn(uint32,uint256,uint256[],bool,uint256,address) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#913-951):
    External calls:
    - (amountsToReceive,trailingFeeLpTokens) = IPool(_poolAddr)._burn(LPTokensToBurn,poolStates[_poolAddr].totalSupplyBase) (contracts/mosaic-alpha-
contracts/Vault/Vault.sol#921)
    - _burnFromInternalBalance(_poolAddr,msg.sender,LPTokensToBurn,true,true) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#938)
        - IMosaicOracle(mosaicOracle).consultUsdPoolPricePerLp(poolAddr,ts) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#703)
        - IPool(token).emitTransfer(from,address(0),amount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#536)
        - IFees(fees).onBeforeTransfer(_pool,_from,_to,_internalBalance[_from][_pool],_internalBalance[_to]
[_pool],amount,originalTotalsupplyBase,trader,payload) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#598-601)
        - performanceFeesToMint = (IPool(_pool).updatePerfFeePricePerLp(tsTmp,valuePerLp)) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#568)
        - IPool(_pool).emitTransfer(address(0),fees,amount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#610)
        - IFees(fees).onBeforeTransfer(_pool,_from,_to,_internalBalance[_from][_pool],_internalBalance[_to][_pool],amount,0,address(0),payload)
(contracts/mosaic-alpha-contracts/Vault/Vault.sol#614)
    Event emitted after the call(s):
    - InternalBalanceChanged(from,token,- int256(amount)) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#535)
        - _burnFromInternalBalance(_poolAddr,msg.sender,LPTokensToBurn,true,true) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#938)
    - InternalBalanceChanged(fees,_pool,int256(uint256(expAmount))) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#609)
        - _burnFromInternalBalance(_poolAddr,msg.sender,LPTokensToBurn,true,true) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#938)
    - TotalSupplyBaseChanged(token,poolStates[token].totalSupplyBase) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#533)
        - _burnFromInternalBalance(_poolAddr,msg.sender,LPTokensToBurn,true,true) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#938)
    - TotalSupplyBaseChanged(_pool,poolStates[_pool].totalSupplyBase) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#605)
        - _burnFromInternalBalance(_poolAddr,msg.sender,LPTokensToBurn,true,true) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#938)
Reentrancy in Vault.Burn(uint32,uint256,uint256[],bool,uint256,address) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#913-951):
    External calls:
    - (amountsToReceive,trailingFeeLpTokens) = IPool(_poolAddr)._burn(LPTokensToBurn,poolStates[_poolAddr].totalSupplyBase) (contracts/mosaic-alpha-
contracts/Vault/Vault.sol#921)
    - (trailing,performance) = _calculateTrailingAndPerformanceFee(_poolAddr,true,false) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#941)
        - IMosaicOracle(mosaicOracle).consultUsdPoolPricePerLp(poolAddr,ts) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#703)
        - performanceFeesToMint = (IPool(_pool).updatePerfFeePricePerLp(tsTmp,valuePerLp)) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#568)
    - IFees(fees).allocateTrailingFee(_poolAddr,trader,trailing,poolStates[_poolAddr].totalSupplyBase.amount,tx.origin) (contracts/mosaic-alpha-
contracts/Vault/Vault.sol#942)
    - IFees(fees).allocatePerformanceFee(_poolAddr,trader,performance,poolStates[_poolAddr].totalSupplyBase.amount,tx.origin) (contracts/mosaic-alpha-
contracts/Vault/Vault.sol#943)
    - _mintToInternalBalance(_poolAddr,fees,trailing + performance,true,true) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#944)
        - IMosaicOracle(mosaicOracle).consultUsdPoolPricePerLp(poolAddr,ts) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#703)
        - IPool(token).emitTransfer(address(0),to,amount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#521)
        - IFees(fees).onBeforeTransfer(_pool,_from,_to,_internalBalance[_from][_pool],_internalBalance[_to]
[_pool],amount,originalTotalsupplyBase,trader,payload) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#598-601)
        - performanceFeesToMint = (IPool(_pool).updatePerfFeePricePerLp(tsTmp,valuePerLp)) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#568)
        - IPool(_pool).emitTransfer(address(0),fees,amount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#610)
        - IFees(fees).onBeforeTransfer(_pool,_from,_to,_internalBalance[_from][_pool],_internalBalance[_to][_pool],amount,0,address(0),payload)
(contracts/mosaic-alpha-contracts/Vault/Vault.sol#614)
    Event emitted after the call(s):
    - InternalBalanceChanged(to,token,int256(amount)) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#520)
        - _mintToInternalBalance(_poolAddr,fees,trailing + performance,true,true) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#944)
    - InternalBalanceChanged(fees,_pool,int256(uint256(expAmount))) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#609)
        - _mintToInternalBalance(_poolAddr,fees,trailing + performance,true,true) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#944)
    - TotalSupplyBaseChanged(token,poolStates[token].totalSupplyBase) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#516)
        - _mintToInternalBalance(_poolAddr,fees,trailing + performance,true,true) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#944)
    - TotalSupplyBaseChanged(_pool,poolStates[_pool].totalSupplyBase) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#605)
        - _mintToInternalBalance(_poolAddr,fees,trailing + performance,true,true) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#944)
Reentrancy in Vault.Burn(uint32,uint256,uint256[],bool,uint256,address) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#913-951):
    External calls:
    - (amountsToReceive,trailingFeeLpTokens) = IPool(_poolAddr)._burn(LPTokensToBurn,poolStates[_poolAddr].totalSupplyBase) (contracts/mosaic-alpha-
contracts/Vault/Vault.sol#921)
    - (trailing,performance) = _calculateTrailingAndPerformanceFee(_poolAddr,true,false) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#941)
        - IMosaicOracle(mosaicOracle).consultUsdPoolPricePerLp(poolAddr,ts) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#703)
        - performanceFeesToMint = (IPool(_pool).updatePerfFeePricePerLp(tsTmp,valuePerLp)) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#568)
    - IFees(fees).allocateTrailingFee(_poolAddr,trader,trailing,poolStates[_poolAddr].totalSupplyBase.amount,tx.origin) (contracts/mosaic-alpha-
contracts/Vault/Vault.sol#942)
    - IFees(fees).allocatePerformanceFee(_poolAddr,trader,performance,poolStates[_poolAddr].totalSupplyBase.amount,tx.origin) (contracts/mosaic-alpha-
contracts/Vault/Vault.sol#943)
```

```
    - _mintToInternalBalance(_poolAddr,fees,trailing + performance,true,true) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#944)
        - IMosaicOracle(mosaicOracle).consultUsdPoolPricePerLp(poolAddr,ts) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#703)
        - IPool(token).emitTransfer(address(0),to,amount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#521)
        - IFees(fees).onBeforeTransfer(_pool,_from,_to,_internalBalance[_from][_pool],_internalBalance[_to]
[_pool],amount,originalTotalsupplyBase,trader,payload) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#598-601)
        - performanceFeesToMint = (IPool(_pool).updatePerfFeePricePerLp(tsTmp,valuePerLp)) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#568)
        - IPool(_pool).emitTransfer(address(0),fees,amount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#610)
        - IFees(fees).onBeforeTransfer(_pool,_from,_to,_internalBalance[_from][_pool],_internalBalance[_to][_pool],amount,0,address(0),payload)
(contracts/mosaic-alpha-contracts/Vault/Vault.sol#614)
    - _burnFromInternalBalance(_poolAddr,msg.sender,LPTokensToBurn,true,true) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#946)
        - IMosaicOracle(mosaicOracle).consultUsdPoolPricePerLp(poolAddr,ts) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#703)
        - IPool(token).emitTransfer(from,address(0),amount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#536)
        - IFees(fees).onBeforeTransfer(_pool,_from,_to,_internalBalance[_from][_pool],_internalBalance[_to]
[_pool],amount,originalTotalsupplyBase,trader,payload) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#598-601)
        - performanceFeesToMint = (IPool(_pool).updatePerfFeePricePerLp(tsTmp,valuePerLp)) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#568)
        - IPool(_pool).emitTransfer(address(0),fees,amount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#610)
        - IFees(fees).onBeforeTransfer(_pool,_from,_to,_internalBalance[_from][_pool],_internalBalance[_to][_pool],amount,0,address(0),payload)
(contracts/mosaic-alpha-contracts/Vault/Vault.sol#614)
    Event emitted after the call(s):
    - InternalBalanceChanged(from,token,- int256(amount)) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#535)
        - _burnFromInternalBalance(_poolAddr,msg.sender,LPTokensToBurn,true,true) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#946)
    - InternalBalanceChanged(fees,_pool,int256(uint256(expAmount))) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#609)
        - _burnFromInternalBalance(_poolAddr,msg.sender,LPTokensToBurn,true,true) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#946)
    - TotalSupplyBaseChanged(token,poolStates[token].totalSupplyBase) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#533)
        - _burnFromInternalBalance(_poolAddr,msg.sender,LPTokensToBurn,true,true) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#946)
    - TotalSupplyBaseChanged(_pool,poolStates[_pool].totalSupplyBase) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#605)
        - _burnFromInternalBalance(_poolAddr,msg.sender,LPTokensToBurn,true,true) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#946)
Reentrancy in Vault.Burn(uint32,uint256,uint256[],bool,uint256,address) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#913-951):
    External calls:
    - (amountsToReceive,trailingFeeLpTokens) = IPool(_poolAddr)._burn(LPTokensToBurn,poolStates[_poolAddr].totalSupplyBase) (contracts/mosaic-alpha-
contracts/Vault/Vault.sol#921)
    - _burnFromInternalBalance(_poolAddr,msg.sender,LPTokensToBurn,true,true) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#938)
        - IMosaicOracle(mosaicOracle).consultUsdPoolPricePerLp(poolAddr,ts) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#703)
        - IPool(token).emitTransfer(from,address(0),amount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#536)
        - IFees(fees).onBeforeTransfer(_pool,_from,_to,_internalBalance[_from][_pool],_internalBalance[_to]
[_pool],amount,originalTotalsupplyBase,trader,payload) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#598-601)
        - performanceFeesToMint = (IPool(_pool).updatePerfFeePricePerLp(tsTmp,valuePerLp)) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#568)
        - IPool(_pool).emitTransfer(address(0),fees,amount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#610)
        - IFees(fees).onBeforeTransfer(_pool,_from,_to,_internalBalance[_from][_pool],_internalBalance[_to][_pool],amount,0,address(0),payload)
(contracts/mosaic-alpha-contracts/Vault/Vault.sol#614)
    - (trailing,performance) = _calculateTrailingAndPerformanceFee(_poolAddr,true,false) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#941)
        - IMosaicOracle(mosaicOracle).consultUsdPoolPricePerLp(poolAddr,ts) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#703)
        - performanceFeesToMint = (IPool(_pool).updatePerfFeePricePerLp(tsTmp,valuePerLp)) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#568)
    - IFees(fees).allocateTrailingFee(_poolAddr,trader,trailing,poolStates[_poolAddr].totalSupplyBase.amount,tx.origin) (contracts/mosaic-alpha-
contracts/Vault/Vault.sol#942)
    - IFees(fees).allocatePerformanceFee(_poolAddr,trader,performance,poolStates[_poolAddr].totalSupplyBase.amount,tx.origin) (contracts/mosaic-alpha-
contracts/Vault/Vault.sol#943)
    - _mintToInternalBalance(_poolAddr,fees,trailing + performance,true,true) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#944)
        - IMosaicOracle(mosaicOracle).consultUsdPoolPricePerLp(poolAddr,ts) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#703)
        - IPool(token).emitTransfer(address(0),to,amount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#521)
        - IFees(fees).onBeforeTransfer(_pool,_from,_to,_internalBalance[_from][_pool],_internalBalance[_to]
[_pool],amount,originalTotalsupplyBase,trader,payload) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#598-601)
        - performanceFeesToMint = (IPool(_pool).updatePerfFeePricePerLp(tsTmp,valuePerLp)) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#568)
        - IPool(_pool).emitTransfer(address(0),fees,amount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#610)
        - IFees(fees).onBeforeTransfer(_pool,_from,_to,_internalBalance[_from][_pool],_internalBalance[_to][_pool],amount,0,address(0),payload)
(contracts/mosaic-alpha-contracts/Vault/Vault.sol#614)
    - _burnFromInternalBalance(_poolAddr,msg.sender,LPTokensToBurn,true,true) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#946)
        - IMosaicOracle(mosaicOracle).consultUsdPoolPricePerLp(poolAddr,ts) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#703)
        - IPool(token).emitTransfer(from,address(0),amount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#536)
        - IFees(fees).onBeforeTransfer(_pool,_from,_to,_internalBalance[_from][_pool],_internalBalance[_to]
[_pool],amount,originalTotalsupplyBase,trader,payload) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#598-601)
        - performanceFeesToMint = (IPool(_pool).updatePerfFeePricePerLp(tsTmp,valuePerLp)) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#568)
        - IPool(_pool).emitTransfer(address(0),fees,amount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#610)
        - IFees(fees).onBeforeTransfer(_pool,_from,_to,_internalBalance[_from][_pool],_internalBalance[_to][_pool],amount,0,address(0),payload)
(contracts/mosaic-alpha-contracts/Vault/Vault.sol#614)
    Event emitted after the call(s):
    - Burned(poolId,msg.sender,from,LPTokensToBurn,tokens,amountsToReceive,toInternalBalance) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#949)
Reentrancy in Vault.Mint(uint32,uint256,uint256[],address,address,bool,uint256,uint256) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#841-904):
    External calls:
    - (amountsToSpend,buyFeeLpTokens,None) = IPool(_poolAddr)._mint(LPTokensRequested,poolStates[_poolAddr].totalSupplyBase) (contracts/mosaic-alpha-
contracts/Vault/Vault.sol#861)
    - _transferInternalBalance(tokens[i],msg.sender,_poolAddr,amountsToSpend[i]) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#872)
        - IMosaicOracle(mosaicOracle).consultUsdPoolPricePerLp(poolAddr,ts) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#703)
        - IFees(fees).onBeforeTransfer(_pool,_from,_to,_internalBalance[_from][_pool],_internalBalance[_to]
[_pool],amount,originalTotalsupplyBase,trader,payload) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#598-601)
        - IPool(token).emitTransfer(from,to,amount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#499)
        - performanceFeesToMint = (IPool(_pool).updatePerfFeePricePerLp(tsTmp,valuePerLp)) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#568)
        - IPool(_pool).emitTransfer(address(0),fees,amount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#610)
        - IFees(fees).onBeforeTransfer(_pool,_from,_to,_internalBalance[_from][_pool],_internalBalance[_to][_pool],amount,0,address(0),payload)
(contracts/mosaic-alpha-contracts/Vault/Vault.sol#614)
    Event emitted after the call(s):
    - InternalBalanceChanged(from,token,- int256(amount)) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#489)
        - _transferInternalBalance(tokens[i],msg.sender,_poolAddr,amountsToSpend[i]) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#872)
    - InternalBalanceChanged(to,token,int256(amount)) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#490)
        - _transferInternalBalance(tokens[i],msg.sender,_poolAddr,amountsToSpend[i]) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#872)
    - InternalBalanceChanged(fees,_pool,int256(uint256(expAmount))) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#609)
        - _transferInternalBalance(tokens[i],msg.sender,_poolAddr,amountsToSpend[i]) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#872)
    - TotalSupplyBaseChanged(_pool,poolStates[_pool].totalSupplyBase) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#605)
        - _transferInternalBalance(tokens[i],msg.sender,_poolAddr,amountsToSpend[i]) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#872)
Reentrancy in Vault.Mint(uint32,uint256,uint256[],address,address,bool,uint256,uint256) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#841-904):
    External calls:
    - (amountsToSpend,buyFeeLpTokens,None) = IPool(_poolAddr)._mint(LPTokensRequested,poolStates[_poolAddr].totalSupplyBase) (contracts/mosaic-alpha-
contracts/Vault/Vault.sol#861)
    - _depositToInternalBalance(tokens[i],msg.sender,_poolAddr,amountsToSpend[i],true) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#876)
        - returndata = address(token).functionCall(data,SafeERC20: low-level call failed)
(node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#110)
        - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#137)
        - IERC20(token).safeTransferFrom(from,address(this),amount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#463)
    External calls sending eth:
    - _depositToInternalBalance(tokens[i],msg.sender,_poolAddr,amountsToSpend[i],true) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#876)
        - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#137)
    Event emitted after the call(s):
    - InternalBalanceChanged(to,token,int256(amount)) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#465)
        - _depositToInternalBalance(tokens[i],msg.sender,_poolAddr,amountsToSpend[i],true) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#876)
Reentrancy in Vault.Mint(uint32,uint256,uint256[],address,address,bool,uint256,uint256) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#841-904):
    External calls:
    - (amountsToSpend,buyFeeLpTokens,None) = IPool(_poolAddr)._mint(LPTokensRequested,poolStates[_poolAddr].totalSupplyBase) (contracts/mosaic-alpha-
contracts/Vault/Vault.sol#861)
    - _registerUser(to,referredBy) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#886)
        - IUserProfile(userProfile).registerUser(to,referredBy) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#202)
    - _mintToInternalBalance(_poolAddr,to,LPTokensRequested,true,true) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#887)
        - IMosaicOracle(mosaicOracle).consultUsdPoolPricePerLp(poolAddr,ts) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#703)
        - IPool(token).emitTransfer(address(0),to,amount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#521)
        - IFees(fees).onBeforeTransfer(_pool,_from,_to,_internalBalance[_from][_pool],_internalBalance[_to]
[_pool],amount,originalTotalsupplyBase,trader,payload) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#598-601)
        - performanceFeesToMint = (IPool(_pool).updatePerfFeePricePerLp(tsTmp,valuePerLp)) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#568)
        - IPool(_pool).emitTransfer(address(0),fees,amount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#610)
```

```
        - IFees(fees).onBeforeTransfer(_pool,_from,_to,_internalBalance[_from][_pool],_internalBalance[_to][_pool],amount,0,address(0),payload)
(contracts/mosaic-alpha-contracts/Vault/Vault.sol#614)
    Event emitted after the call(s):
    - InternalBalanceChanged(to,token,int256(amount)) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#520)
        - _mintToInternalBalance(_poolAddr,to,LPTokensRequested,true,true) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#887)
    - InternalBalanceChanged(fees,_pool,int256(uint256(expAmount))) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#609)
        - _mintToInternalBalance(_poolAddr,to,LPTokensRequested,true,true) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#887)
    - TotalSupplyBaseChanged(token,poolStates[token].totalSupplyBase) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#516)
        - _mintToInternalBalance(_poolAddr,to,LPTokensRequested,true,true) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#887)
    - TotalSupplyBaseChanged(_pool,poolStates[_pool].totalSupplyBase) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#605)
        - _mintToInternalBalance(_poolAddr,to,LPTokensRequested,true,true) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#887)
Reentrancy in Vault.Mint(uint32,uint256,uint256[],address,address,bool,uint256,uint256) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#841-904):
    External calls:
    - (amountsToSpend,buyFeeLpTokens,None) = IPool(_poolAddr)._mint(LPTokensRequested,poolStates[_poolAddr].totalSupplyBase) (contracts/mosaic-alpha-
contracts/Vault/Vault.sol#861)
    - _registerUserPurchase(to,referredBy,trader,tokens,amountsToSpend,usdValue) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#891)
        - IAffiliate(affiliate).registerUserPurchase(to,referredBy,trader,usdAmount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#207)
        - IAffiliate(affiliate).registerUserPurchaseAsTokens(to,referredBy,trader,tokens,tokenAmounts) (contracts/mosaic-alpha-
contracts/Vault/Vault.sol#209)
    - (trailing,performance) = _calculateTrailingAndPerformanceFee(_poolAddr,true,false) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#893)
        - IMosaicOracle(mosaicOracle).consultUsdPoolPricePerLp(poolAddr,ts) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#703)
        - performanceFeesToMint = (IPool(_pool).updatePerfFeePricePerLp(tsTmp,valuePerLp)) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#568)
    - IFees(fees).allocateTrailingFee(_poolAddr,trader,trailing,poolStates[_poolAddr].totalSupplyBase.amount,tx.origin) (contracts/mosaic-alpha-
contracts/Vault/Vault.sol#894)
    - IFees(fees).allocatePerformanceFee(_poolAddr,trader,performance,poolStates[_poolAddr].totalSupplyBase.amount,tx.origin) (contracts/mosaic-alpha-
contracts/Vault/Vault.sol#895)
    - _mintToInternalBalance(_poolAddr,fees,trailing + performance + buyFeeLpTokens,true,true) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#896)
        - IMosaicOracle(mosaicOracle).consultUsdPoolPricePerLp(poolAddr,ts) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#703)
        - IPool(token).emitTransfer(address(0),to,amount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#521)
        - IFees(fees).onBeforeTransfer(_pool,_from,_to,_internalBalance[_from][_pool],_internalBalance[_to]
[_pool],amount,originalTotalsupplyBase,trader,payload) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#598-601)
        - performanceFeesToMint = (IPool(_pool).updatePerfFeePricePerLp(tsTmp,valuePerLp)) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#568)
        - IPool(_pool).emitTransfer(address(0),fees,amount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#610)
        - IFees(fees).onBeforeTransfer(_pool,_from,_to,_internalBalance[_from][_pool],_internalBalance[_to][_pool],amount,0,address(0),payload)
(contracts/mosaic-alpha-contracts/Vault/Vault.sol#614)
    Event emitted after the call(s):
    - InternalBalanceChanged(to,token,int256(amount)) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#520)
        - _mintToInternalBalance(_poolAddr,fees,trailing + performance + buyFeeLpTokens,true,true) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#896)
    - InternalBalanceChanged(fees,_pool,int256(uint256(expAmount))) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#609)
        - _mintToInternalBalance(_poolAddr,fees,trailing + performance + buyFeeLpTokens,true,true) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#896)
    - TotalSupplyBaseChanged(token,poolStates[token].totalSupplyBase) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#516)
        - _mintToInternalBalance(_poolAddr,fees,trailing + performance + buyFeeLpTokens,true,true) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#896)
    - TotalSupplyBaseChanged(_pool,poolStates[_pool].totalSupplyBase) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#605)
        - _mintToInternalBalance(_poolAddr,fees,trailing + performance + buyFeeLpTokens,true,true) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#896)
Reentrancy in Vault.Mint(uint32,uint256,uint256[],address,address,bool,uint256,uint256) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#841-904):
    External calls:
    - (amountsToSpend,buyFeeLpTokens,None) = IPool(_poolAddr)._mint(LPTokensRequested,poolStates[_poolAddr].totalSupplyBase) (contracts/mosaic-alpha-
contracts/Vault/Vault.sol#861)
    - _registerUserPurchase(to,referredBy,trader,tokens,amountsToSpend,usdValue) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#891)
        - IAffiliate(affiliate).registerUserPurchase(to,referredBy,trader,usdAmount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#207)
        - IAffiliate(affiliate).registerUserPurchaseAsTokens(to,referredBy,trader,tokens,tokenAmounts) (contracts/mosaic-alpha-
contracts/Vault/Vault.sol#209)
    - (trailing,performance) = _calculateTrailingAndPerformanceFee(_poolAddr,true,false) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#893)
        - IMosaicOracle(mosaicOracle).consultUsdPoolPricePerLp(poolAddr,ts) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#703)
        - performanceFeesToMint = (IPool(_pool).updatePerfFeePricePerLp(tsTmp,valuePerLp)) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#568)
    - IFees(fees).allocateTrailingFee(_poolAddr,trader,trailing,poolStates[_poolAddr].totalSupplyBase.amount,tx.origin) (contracts/mosaic-alpha-
contracts/Vault/Vault.sol#894)
    - IFees(fees).allocatePerformanceFee(_poolAddr,trader,performance,poolStates[_poolAddr].totalSupplyBase.amount,tx.origin) (contracts/mosaic-alpha-
contracts/Vault/Vault.sol#895)
    - _mintToInternalBalance(_poolAddr,fees,trailing + performance + buyFeeLpTokens,true,true) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#896)
        - IMosaicOracle(mosaicOracle).consultUsdPoolPricePerLp(poolAddr,ts) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#703)
        - IPool(token).emitTransfer(address(0),to,amount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#521)
        - IFees(fees).onBeforeTransfer(_pool,_from,_to,_internalBalance[_from][_pool],_internalBalance[_to]
[_pool],amount,originalTotalsupplyBase,trader,payload) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#598-601)
        - performanceFeesToMint = (IPool(_pool).updatePerfFeePricePerLp(tsTmp,valuePerLp)) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#568)
        - IPool(_pool).emitTransfer(address(0),fees,amount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#610)
        - IFees(fees).onBeforeTransfer(_pool,_from,_to,_internalBalance[_from][_pool],_internalBalance[_to][_pool],amount,0,address(0),payload)
(contracts/mosaic-alpha-contracts/Vault/Vault.sol#614)
    - IFees(fees).allocateBuyFee(_poolAddr,to,trader,buyFeeLpTokens) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#898)
    - _mintToInternalBalance(_poolAddr,to,LPTokensRequested,true,true) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#899)
        - IMosaicOracle(mosaicOracle).consultUsdPoolPricePerLp(poolAddr,ts) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#703)
        - IPool(token).emitTransfer(address(0),to,amount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#521)
        - IFees(fees).onBeforeTransfer(_pool,_from,_to,_internalBalance[_from][_pool],_internalBalance[_to]
[_pool],amount,originalTotalsupplyBase,trader,payload) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#598-601)
        - performanceFeesToMint = (IPool(_pool).updatePerfFeePricePerLp(tsTmp,valuePerLp)) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#568)
        - IPool(_pool).emitTransfer(address(0),fees,amount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#610)
        - IFees(fees).onBeforeTransfer(_pool,_from,_to,_internalBalance[_from][_pool],_internalBalance[_to][_pool],amount,0,address(0),payload)
(contracts/mosaic-alpha-contracts/Vault/Vault.sol#614)
    Event emitted after the call(s):
    - InternalBalanceChanged(to,token,int256(amount)) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#520)
        - _mintToInternalBalance(_poolAddr,to,LPTokensRequested,true,true) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#899)
    - InternalBalanceChanged(fees,_pool,int256(uint256(expAmount))) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#609)
        - _mintToInternalBalance(_poolAddr,to,LPTokensRequested,true,true) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#899)
    - TotalSupplyBaseChanged(token,poolStates[token].totalSupplyBase) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#516)
        - _mintToInternalBalance(_poolAddr,to,LPTokensRequested,true,true) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#899)
    - TotalSupplyBaseChanged(_pool,poolStates[_pool].totalSupplyBase) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#605)
        - _mintToInternalBalance(_poolAddr,to,LPTokensRequested,true,true) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#899)
Reentrancy in Vault.Mint(uint32,uint256,uint256[],address,address,bool,uint256,uint256) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#841-904):
    External calls:
    - (amountsToSpend,buyFeeLpTokens,None) = IPool(_poolAddr)._mint(LPTokensRequested,poolStates[_poolAddr].totalSupplyBase) (contracts/mosaic-alpha-
contracts/Vault/Vault.sol#861)
    - _registerUser(to,referredBy) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#886)
        - IUserProfile(userProfile).registerUser(to,referredBy) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#202)
    - _mintToInternalBalance(_poolAddr,to,LPTokensRequested,true,true) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#887)
        - IMosaicOracle(mosaicOracle).consultUsdPoolPricePerLp(poolAddr,ts) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#703)
        - IPool(token).emitTransfer(address(0),to,amount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#521)
        - IFees(fees).onBeforeTransfer(_pool,_from,_to,_internalBalance[_from][_pool],_internalBalance[_to]
[_pool],amount,originalTotalsupplyBase,trader,payload) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#598-601)
        - performanceFeesToMint = (IPool(_pool).updatePerfFeePricePerLp(tsTmp,valuePerLp)) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#568)
        - IPool(_pool).emitTransfer(address(0),fees,amount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#610)
        - IFees(fees).onBeforeTransfer(_pool,_from,_to,_internalBalance[_from][_pool],_internalBalance[_to][_pool],amount,0,address(0),payload)
(contracts/mosaic-alpha-contracts/Vault/Vault.sol#614)
    - _registerUserPurchase(to,referredBy,trader,tokens,amountsToSpend,usdValue) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#891)
        - IAffiliate(affiliate).registerUserPurchase(to,referredBy,trader,usdAmount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#207)
        - IAffiliate(affiliate).registerUserPurchaseAsTokens(to,referredBy,trader,tokens,tokenAmounts) (contracts/mosaic-alpha-
contracts/Vault/Vault.sol#209)
    - (trailing,performance) = _calculateTrailingAndPerformanceFee(_poolAddr,true,false) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#893)
        - IMosaicOracle(mosaicOracle).consultUsdPoolPricePerLp(poolAddr,ts) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#703)
        - performanceFeesToMint = (IPool(_pool).updatePerfFeePricePerLp(tsTmp,valuePerLp)) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#568)
    - IFees(fees).allocateTrailingFee(_poolAddr,trader,trailing,poolStates[_poolAddr].totalSupplyBase.amount,tx.origin) (contracts/mosaic-alpha-
contracts/Vault/Vault.sol#894)
    - IFees(fees).allocatePerformanceFee(_poolAddr,trader,performance,poolStates[_poolAddr].totalSupplyBase.amount,tx.origin) (contracts/mosaic-alpha-
contracts/Vault/Vault.sol#895)
    - _mintToInternalBalance(_poolAddr,fees,trailing + performance + buyFeeLpTokens,true,true) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#896)
        - IMosaicOracle(mosaicOracle).consultUsdPoolPricePerLp(poolAddr,ts) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#703)
        - IPool(token).emitTransfer(address(0),to,amount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#521)
        - IFees(fees).onBeforeTransfer(_pool,_from,_to,_internalBalance[_from][_pool],_internalBalance[_to]
```

```
[_pool],amount,originalTotalsupplyBase,trader,payload) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#598-601)
        - performanceFeesToMint = (IPool(_pool).updatePerfFeePricePerLp(tsTmp,valuePerLp)) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#568)
        - IPool(_pool).emitTransfer(address(0),fees,amount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#610)
        - IFees(fees).onBeforeTransfer(_pool,_from,_to,_internalBalance[_from][_pool],_internalBalance[_to][_pool],amount,0,address(0),payload)
(contracts/mosaic-alpha-contracts/Vault/Vault.sol#614)
    - IFees(fees).allocateBuyFee(_poolAddr,to,trader,buyFeeLpTokens) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#898)
    - _mintToInternalBalance(_poolAddr,to,LPTokensRequested,true,true) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#899)
        - IMosaicOracle(mosaicOracle).consultUsdPoolPricePerLp(poolAddr,ts) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#703)
        - IPool(token).emitTransfer(address(0),to,amount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#521)
        - IFees(fees).onBeforeTransfer(_pool,_from,_to,_internalBalance[_from][_pool],_internalBalance[_to]
[_pool],amount,originalTotalsupplyBase,trader,payload) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#598-601)
        - performanceFeesToMint = (IPool(_pool).updatePerfFeePricePerLp(tsTmp,valuePerLp)) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#568)
        - IPool(_pool).emitTransfer(address(0),fees,amount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#610)
        - IFees(fees).onBeforeTransfer(_pool,_from,_to,_internalBalance[_from][_pool],_internalBalance[_to][_pool],amount,0,address(0),payload)
(contracts/mosaic-alpha-contracts/Vault/Vault.sol#614)
    Event emitted after the call(s):
    - Minted(poolId,msg.sender,to,LPTokensRequested,tokens,amountsToSpend,fromInternalBalance) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#902)
Reentrancy in Vault._burnFromInternalBalance(address,address,uint256,bool,bool) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#526-537):
    External calls:
    - _onBeforeBalanceTransfer(token,from,address(0),amount,feesAlreadyAllocated,true,false) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#529)
        - IMosaicOracle(mosaicOracle).consultUsdPoolPricePerLp(poolAddr,ts) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#703)
        - IFees(fees).onBeforeTransfer(_pool,_from,_to,_internalBalance[_from][_pool],_internalBalance[_to]
[_pool],amount,originalTotalsupplyBase,trader,payload) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#598-601)
        - performanceFeesToMint = (IPool(_pool).updatePerfFeePricePerLp(tsTmp,valuePerLp)) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#568)
        - IPool(_pool).emitTransfer(address(0),fees,amount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#610)
        - IFees(fees).onBeforeTransfer(_pool,_from,_to,_internalBalance[_from][_pool],_internalBalance[_to][_pool],amount,0,address(0),payload)
(contracts/mosaic-alpha-contracts/Vault/Vault.sol#614)
    Event emitted after the call(s):
    - InternalBalanceChanged(from,token,- int256(amount)) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#535)
    - TotalSupplyBaseChanged(token,poolStates[token].totalSupplyBase) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#533)
Reentrancy in Vault._depositToInternalBalance(address,address,address,uint256,bool) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#459-466):
    External calls:
    - IERC20(token).safeTransferFrom(from,address(this),amount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#463)
    Event emitted after the call(s):
    - InternalBalanceChanged(to,token,int256(amount)) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#465)
Reentrancy in Vault._mintToInternalBalance(address,address,uint256,bool,bool) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#510-522):
    External calls:
    - _onBeforeBalanceTransfer(token,address(0),to,amount,feesAlreadyAllocated,true,false) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#513)
        - IMosaicOracle(mosaicOracle).consultUsdPoolPricePerLp(poolAddr,ts) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#703)
        - IFees(fees).onBeforeTransfer(_pool,_from,_to,_internalBalance[_from][_pool],_internalBalance[_to]
[_pool],amount,originalTotalsupplyBase,trader,payload) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#598-601)
        - performanceFeesToMint = (IPool(_pool).updatePerfFeePricePerLp(tsTmp,valuePerLp)) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#568)
        - IPool(_pool).emitTransfer(address(0),fees,amount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#610)
        - IFees(fees).onBeforeTransfer(_pool,_from,_to,_internalBalance[_from][_pool],_internalBalance[_to][_pool],amount,0,address(0),payload)
(contracts/mosaic-alpha-contracts/Vault/Vault.sol#614)
    Event emitted after the call(s):
    - InternalBalanceChanged(to,token,int256(amount)) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#520)
    - TotalSupplyBaseChanged(token,poolStates[token].totalSupplyBase) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#516)
Reentrancy in Vault._onBeforeBalanceTransfer(address,address,address,uint256,bool,bool,bool) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#583-615):
    External calls:
    - (payload.trailingLpToMint,payload.performanceLpToMint) =
_calculateTrailingAndPerformanceFee(_pool,forceTrailingFeeAllocation,forcePerformanceFeeAllocation) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#591)
        - IMosaicOracle(mosaicOracle).consultUsdPoolPricePerLp(poolAddr,ts) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#703)
        - performanceFeesToMint = (IPool(_pool).updatePerfFeePricePerLp(tsTmp,valuePerLp)) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#568)
    - IFees(fees).onBeforeTransfer(_pool,_from,_to,_internalBalance[_from][_pool],_internalBalance[_to]
[_pool],amount,originalTotalsupplyBase,trader,payload) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#598-601)
    Event emitted after the call(s):
    - InternalBalanceChanged(fees,_pool,int256(uint256(expAmount))) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#609)
    - TotalSupplyBaseChanged(_pool,poolStates[_pool].totalSupplyBase) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#605)
Reentrancy in Vault._withdrawFromInternalBalance(address,address,address,uint256) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#470-479):
    External calls:
    - IERC20(token).safeTransfer(to,amount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#476)
    Event emitted after the call(s):
    - InternalBalanceChanged(from,token,- int256(amount)) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#478)
Reentrancy in Vault.addInitialLiquidity(uint32,address[],uint256[],address,bool) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#779-815):
    External calls:
    - (lpTokens,dust) = IPool(_poolAddress).initialLiquidityProvided(_tokens,_liquidity) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#806)
    - _mintToInternalBalance(_poolAddress,_lpTo,lpTokens,true,true) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#812)
        - IMosaicOracle(mosaicOracle).consultUsdPoolPricePerLp(poolAddr,ts) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#703)
        - IPool(token).emitTransfer(address(0),to,amount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#521)
        - IFees(fees).onBeforeTransfer(_pool,_from,_to,_internalBalance[_from][_pool],_internalBalance[_to]
[_pool],amount,originalTotalsupplyBase,trader,payload) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#598-601)
        - performanceFeesToMint = (IPool(_pool).updatePerfFeePricePerLp(tsTmp,valuePerLp)) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#568)
        - IPool(_pool).emitTransfer(address(0),fees,amount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#610)
        - IFees(fees).onBeforeTransfer(_pool,_from,_to,_internalBalance[_from][_pool],_internalBalance[_to][_pool],amount,0,address(0),payload)
(contracts/mosaic-alpha-contracts/Vault/Vault.sol#614)
    Event emitted after the call(s):
    - InternalBalanceChanged(to,token,int256(amount)) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#520)
        - _mintToInternalBalance(_poolAddress,_lpTo,lpTokens,true,true) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#812)
    - InternalBalanceChanged(fees,_pool,int256(uint256(expAmount))) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#609)
        - _mintToInternalBalance(_poolAddress,_lpTo,lpTokens,true,true) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#812)
    - TotalSupplyBaseChanged(token,poolStates[token].totalSupplyBase) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#516)
        - _mintToInternalBalance(_poolAddress,_lpTo,lpTokens,true,true) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#812)
    - TotalSupplyBaseChanged(_pool,poolStates[_pool].totalSupplyBase) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#605)
        - _mintToInternalBalance(_poolAddress,_lpTo,lpTokens,true,true) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#812)
Reentrancy in Vault.addInitialLiquidity(uint32,address[],uint256[],address,bool) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#779-815):
    External calls:
    - (lpTokens,dust) = IPool(_poolAddress).initialLiquidityProvided(_tokens,_liquidity) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#806)
    - _mintToInternalBalance(_poolAddress,_lpTo,lpTokens,true,true) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#812)
        - IMosaicOracle(mosaicOracle).consultUsdPoolPricePerLp(poolAddr,ts) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#703)
        - IPool(token).emitTransfer(address(0),to,amount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#521)
        - IFees(fees).onBeforeTransfer(_pool,_from,_to,_internalBalance[_from][_pool],_internalBalance[_to]
[_pool],amount,originalTotalsupplyBase,trader,payload) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#598-601)
        - performanceFeesToMint = (IPool(_pool).updatePerfFeePricePerLp(tsTmp,valuePerLp)) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#568)
        - IPool(_pool).emitTransfer(address(0),fees,amount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#610)
        - IFees(fees).onBeforeTransfer(_pool,_from,_to,_internalBalance[_from][_pool],_internalBalance[_to][_pool],amount,0,address(0),payload)
(contracts/mosaic-alpha-contracts/Vault/Vault.sol#614)
    - _mintToInternalBalance(_poolAddress,address(0),dust,true,true) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#814)
        - IMosaicOracle(mosaicOracle).consultUsdPoolPricePerLp(poolAddr,ts) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#703)
        - IPool(token).emitTransfer(address(0),to,amount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#521)
        - IFees(fees).onBeforeTransfer(_pool,_from,_to,_internalBalance[_from][_pool],_internalBalance[_to]
[_pool],amount,originalTotalsupplyBase,trader,payload) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#598-601)
        - performanceFeesToMint = (IPool(_pool).updatePerfFeePricePerLp(tsTmp,valuePerLp)) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#568)
        - IPool(_pool).emitTransfer(address(0),fees,amount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#610)
        - IFees(fees).onBeforeTransfer(_pool,_from,_to,_internalBalance[_from][_pool],_internalBalance[_to][_pool],amount,0,address(0),payload)
(contracts/mosaic-alpha-contracts/Vault/Vault.sol#614)
    Event emitted after the call(s):
    - InternalBalanceChanged(to,token,int256(amount)) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#520)
        - _mintToInternalBalance(_poolAddress,address(0),dust,true,true) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#814)
    - InternalBalanceChanged(fees,_pool,int256(uint256(expAmount))) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#609)
        - _mintToInternalBalance(_poolAddress,address(0),dust,true,true) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#814)
    - TotalSupplyBaseChanged(token,poolStates[token].totalSupplyBase) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#516)
        - _mintToInternalBalance(_poolAddress,address(0),dust,true,true) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#814)
    - TotalSupplyBaseChanged(_pool,poolStates[_pool].totalSupplyBase) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#605)
        - _mintToInternalBalance(_poolAddress,address(0),dust,true,true) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#814)
Reentrancy in Vault.bid(uint256,uint256,bool,bool,uint256) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1163-1213):
    External calls:
```

```
    - _transferInternalBalance(_auctionDetails.tokenToBuy,msg.sender,_auctionDetails.poolAddress,_toPay) (contracts/mosaic-alpha-
contracts/Vault/Vault.sol#1199)
        - IMosaicOracle(mosaicOracle).consultUsdPoolPricePerLp(poolAddr,ts) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#703)
        - IFees(fees).onBeforeTransfer(_pool,_from,_to,_internalBalance[_from][_pool],_internalBalance[_to]
[_pool],amount,originalTotalsupplyBase,trader,payload) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#598-601)
        - IPool(token).emitTransfer(from,to,amount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#499)
        - performanceFeesToMint = (IPool(_pool).updatePerfFeePricePerLp(tsTmp,valuePerLp)) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#568)
        - IPool(_pool).emitTransfer(address(0),fees,amount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#610)
        - IFees(fees).onBeforeTransfer(_pool,_from,_to,_internalBalance[_from][_pool],_internalBalance[_to][_pool],amount,0,address(0),payload)
(contracts/mosaic-alpha-contracts/Vault/Vault.sol#614)
    - _transferInternalBalance(_auctionDetails.tokenToSell,_auctionDetails.poolAddress,msg.sender,amount) (contracts/mosaic-alpha-
contracts/Vault/Vault.sol#1201)
        - IMosaicOracle(mosaicOracle).consultUsdPoolPricePerLp(poolAddr,ts) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#703)
        - IFees(fees).onBeforeTransfer(_pool,_from,_to,_internalBalance[_from][_pool],_internalBalance[_to]
[_pool],amount,originalTotalsupplyBase,trader,payload) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#598-601)
        - IPool(token).emitTransfer(from,to,amount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#499)
        - performanceFeesToMint = (IPool(_pool).updatePerfFeePricePerLp(tsTmp,valuePerLp)) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#568)
        - IPool(_pool).emitTransfer(address(0),fees,amount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#610)
        - IFees(fees).onBeforeTransfer(_pool,_from,_to,_internalBalance[_from][_pool],_internalBalance[_to][_pool],amount,0,address(0),payload)
(contracts/mosaic-alpha-contracts/Vault/Vault.sol#614)
    Event emitted after the call(s):
    - InternalBalanceChanged(from,token,- int256(amount)) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#489)
        - _transferInternalBalance(_auctionDetails.tokenToSell,_auctionDetails.poolAddress,msg.sender,amount) (contracts/mosaic-alpha-
contracts/Vault/Vault.sol#1201)
    - InternalBalanceChanged(to,token,int256(amount)) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#490)
        - _transferInternalBalance(_auctionDetails.tokenToSell,_auctionDetails.poolAddress,msg.sender,amount) (contracts/mosaic-alpha-
contracts/Vault/Vault.sol#1201)
    - InternalBalanceChanged(fees,_pool,int256(uint256(expAmount))) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#609)
        - _transferInternalBalance(_auctionDetails.tokenToSell,_auctionDetails.poolAddress,msg.sender,amount) (contracts/mosaic-alpha-
contracts/Vault/Vault.sol#1201)
    - TotalSupplyBaseChanged(_pool,poolStates[_pool].totalSupplyBase) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#605)
        - _transferInternalBalance(_auctionDetails.tokenToSell,_auctionDetails.poolAddress,msg.sender,amount) (contracts/mosaic-alpha-
contracts/Vault/Vault.sol#1201)
Reentrancy in Vault.bid(uint256,uint256,bool,bool,uint256) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1163-1213):
    External calls:
    - _depositToInternalBalance(_auctionDetails.tokenToBuy,msg.sender,_auctionDetails.poolAddress,_toPay,false) (contracts/mosaic-alpha-
contracts/Vault/Vault.sol#1203)
        - returndata = address(token).functionCall(data,SafeERC20: low-level call failed)
(node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#110)
        - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#137)
        - IERC20(token).safeTransferFrom(from,address(this),amount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#463)
    - _withdrawFromInternalBalance(_auctionDetails.tokenToSell,_auctionDetails.poolAddress,msg.sender,amount) (contracts/mosaic-alpha-
contracts/Vault/Vault.sol#1204)
        - returndata = address(token).functionCall(data,SafeERC20: low-level call failed)
(node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#110)
        - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#137)
        - IERC20(token).safeTransfer(to,amount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#476)
    External calls sending eth:
    - _depositToInternalBalance(_auctionDetails.tokenToBuy,msg.sender,_auctionDetails.poolAddress,_toPay,false) (contracts/mosaic-alpha-
contracts/Vault/Vault.sol#1203)
        - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#137)
    - _withdrawFromInternalBalance(_auctionDetails.tokenToSell,_auctionDetails.poolAddress,msg.sender,amount) (contracts/mosaic-alpha-
contracts/Vault/Vault.sol#1204)
        - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#137)
    Event emitted after the call(s):
    - InternalBalanceChanged(from,token,- int256(amount)) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#478)
        - _withdrawFromInternalBalance(_auctionDetails.tokenToSell,_auctionDetails.poolAddress,msg.sender,amount) (contracts/mosaic-alpha-
contracts/Vault/Vault.sol#1204)
Reentrancy in Vault.bid(uint256,uint256,bool,bool,uint256) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1163-1213):
    External calls:
    - _transferInternalBalance(_auctionDetails.tokenToBuy,msg.sender,_auctionDetails.poolAddress,_toPay) (contracts/mosaic-alpha-
contracts/Vault/Vault.sol#1199)
        - IMosaicOracle(mosaicOracle).consultUsdPoolPricePerLp(poolAddr,ts) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#703)
        - IFees(fees).onBeforeTransfer(_pool,_from,_to,_internalBalance[_from][_pool],_internalBalance[_to]
[_pool],amount,originalTotalsupplyBase,trader,payload) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#598-601)
        - IPool(token).emitTransfer(from,to,amount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#499)
        - performanceFeesToMint = (IPool(_pool).updatePerfFeePricePerLp(tsTmp,valuePerLp)) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#568)
        - IPool(_pool).emitTransfer(address(0),fees,amount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#610)
        - IFees(fees).onBeforeTransfer(_pool,_from,_to,_internalBalance[_from][_pool],_internalBalance[_to][_pool],amount,0,address(0),payload)
(contracts/mosaic-alpha-contracts/Vault/Vault.sol#614)
    - _transferInternalBalance(_auctionDetails.tokenToSell,_auctionDetails.poolAddress,msg.sender,amount) (contracts/mosaic-alpha-
contracts/Vault/Vault.sol#1201)
        - IMosaicOracle(mosaicOracle).consultUsdPoolPricePerLp(poolAddr,ts) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#703)
        - IFees(fees).onBeforeTransfer(_pool,_from,_to,_internalBalance[_from][_pool],_internalBalance[_to]
[_pool],amount,originalTotalsupplyBase,trader,payload) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#598-601)
        - IPool(token).emitTransfer(from,to,amount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#499)
        - performanceFeesToMint = (IPool(_pool).updatePerfFeePricePerLp(tsTmp,valuePerLp)) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#568)
        - IPool(_pool).emitTransfer(address(0),fees,amount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#610)
        - IFees(fees).onBeforeTransfer(_pool,_from,_to,_internalBalance[_from][_pool],_internalBalance[_to][_pool],amount,0,address(0),payload)
(contracts/mosaic-alpha-contracts/Vault/Vault.sol#614)
    - _depositToInternalBalance(_auctionDetails.tokenToBuy,msg.sender,_auctionDetails.poolAddress,_toPay,false) (contracts/mosaic-alpha-
contracts/Vault/Vault.sol#1203)
        - returndata = address(token).functionCall(data,SafeERC20: low-level call failed)
(node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#110)
        - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#137)
        - IERC20(token).safeTransferFrom(from,address(this),amount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#463)
    - _withdrawFromInternalBalance(_auctionDetails.tokenToSell,_auctionDetails.poolAddress,msg.sender,amount) (contracts/mosaic-alpha-
contracts/Vault/Vault.sol#1204)
        - returndata = address(token).functionCall(data,SafeERC20: low-level call failed)
(node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#110)
        - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#137)
        - IERC20(token).safeTransfer(to,amount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#476)
    External calls sending eth:
    - _depositToInternalBalance(_auctionDetails.tokenToBuy,msg.sender,_auctionDetails.poolAddress,_toPay,false) (contracts/mosaic-alpha-
contracts/Vault/Vault.sol#1203)
        - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#137)
    - _withdrawFromInternalBalance(_auctionDetails.tokenToSell,_auctionDetails.poolAddress,msg.sender,amount) (contracts/mosaic-alpha-
contracts/Vault/Vault.sol#1204)
        - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#137)
    Event emitted after the call(s):
    - NewBid(auctionId,msg.sender,amount,_toPay,_auctionDetails.tokenToBuy,_auctionDetails.tokenToSell,_auctionDetails.remainingAmount) (contracts/mosaic-
alpha-contracts/Vault/Vault.sol#1210)
Reentrancy in Vault.deregisterToken(address,uint256) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#818-831):
    External calls:
    - _transferInternalBalance(_tokenAddress,msg.sender,vaultState.feeTo,_remainingAmount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#828)
        - IMosaicOracle(mosaicOracle).consultUsdPoolPricePerLp(poolAddr,ts) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#703)
        - IFees(fees).onBeforeTransfer(_pool,_from,_to,_internalBalance[_from][_pool],_internalBalance[_to]
[_pool],amount,originalTotalsupplyBase,trader,payload) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#598-601)
        - IPool(token).emitTransfer(from,to,amount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#499)
        - performanceFeesToMint = (IPool(_pool).updatePerfFeePricePerLp(tsTmp,valuePerLp)) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#568)
        - IPool(_pool).emitTransfer(address(0),fees,amount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#610)
        - IFees(fees).onBeforeTransfer(_pool,_from,_to,_internalBalance[_from][_pool],_internalBalance[_to][_pool],amount,0,address(0),payload)
(contracts/mosaic-alpha-contracts/Vault/Vault.sol#614)
    Event emitted after the call(s):
    - TokensDeregistered(_poolId,_tokenAddress) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#830)
Reentrancy in Vault.registerPool(address,address,address) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#721-739):
    External calls:
```

```
    - _registerUser(_user,_referredBy) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#727)
        - IUserProfile(userProfile).registerUser(to,referredBy) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#202)
    Event emitted after the call(s):
    - PoolRegistered(_poolId,_pool) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#737)
Reentrancy in Vault.swap(address,bool,address,address,uint256,bool,uint256,uint64) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#969-1032):
    External calls:
    - _transferInternalBalance(tokenIn,msg.sender,poolAddress,_amountIn) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1010)
        - IMosaicOracle(mosaicOracle).consultUsdPoolPricePerLp(poolAddr,ts) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#703)
        - IFees(fees).onBeforeTransfer(_pool,_from,_to,_internalBalance[_from][_pool],_internalBalance[_to]
[_pool],amount,originalTotalsupplyBase,trader,payload) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#598-601)
        - IPool(token).emitTransfer(from,to,amount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#499)
        - performanceFeesToMint = (IPool(_pool).updatePerfFeePricePerLp(tsTmp,valuePerLp)) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#568)
        - IPool(_pool).emitTransfer(address(0),fees,amount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#610)
        - IFees(fees).onBeforeTransfer(_pool,_from,_to,_internalBalance[_from][_pool],_internalBalance[_to][_pool],amount,0,address(0),payload)
(contracts/mosaic-alpha-contracts/Vault/Vault.sol#614)
    - _transferInternalBalance(tokenOut,poolAddress,msg.sender,_amountOut) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1012)
        - IMosaicOracle(mosaicOracle).consultUsdPoolPricePerLp(poolAddr,ts) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#703)
        - IFees(fees).onBeforeTransfer(_pool,_from,_to,_internalBalance[_from][_pool],_internalBalance[_to]
[_pool],amount,originalTotalsupplyBase,trader,payload) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#598-601)
        - IPool(token).emitTransfer(from,to,amount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#499)
        - performanceFeesToMint = (IPool(_pool).updatePerfFeePricePerLp(tsTmp,valuePerLp)) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#568)
        - IPool(_pool).emitTransfer(address(0),fees,amount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#610)
        - IFees(fees).onBeforeTransfer(_pool,_from,_to,_internalBalance[_from][_pool],_internalBalance[_to][_pool],amount,0,address(0),payload)
(contracts/mosaic-alpha-contracts/Vault/Vault.sol#614)
    Event emitted after the call(s):
    - InternalBalanceChanged(from,token,- int256(amount)) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#489)
        - _transferInternalBalance(tokenOut,poolAddress,msg.sender,_amountOut) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1012)
    - InternalBalanceChanged(to,token,int256(amount)) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#490)
        - _transferInternalBalance(tokenOut,poolAddress,msg.sender,_amountOut) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1012)
    - InternalBalanceChanged(fees,_pool,int256(uint256(expAmount))) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#609)
        - _transferInternalBalance(tokenOut,poolAddress,msg.sender,_amountOut) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1012)
    - TotalSupplyBaseChanged(_pool,poolStates[_pool].totalSupplyBase) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#605)
        - _transferInternalBalance(tokenOut,poolAddress,msg.sender,_amountOut) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1012)
Reentrancy in Vault.swap(address,bool,address,address,uint256,bool,uint256,uint64) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#969-1032):
    External calls:
    - _depositToInternalBalance(tokenIn,msg.sender,poolAddress,_amountIn,true) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1017)
        - returndata = address(token).functionCall(data,SafeERC20: low-level call failed)
(node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#110)
        - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#137)
        - IERC20(token).safeTransferFrom(from,address(this),amount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#463)
    - _withdrawFromInternalBalance(tokenOut,poolAddress,msg.sender,_amountOut) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1019)
        - returndata = address(token).functionCall(data,SafeERC20: low-level call failed)
(node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#110)
        - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#137)
        - IERC20(token).safeTransfer(to,amount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#476)
    External calls sending eth:
    - _depositToInternalBalance(tokenIn,msg.sender,poolAddress,_amountIn,true) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1017)
        - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#137)
    - _withdrawFromInternalBalance(tokenOut,poolAddress,msg.sender,_amountOut) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1019)
        - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#137)
    Event emitted after the call(s):
    - InternalBalanceChanged(from,token,- int256(amount)) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#478)
        - _withdrawFromInternalBalance(tokenOut,poolAddress,msg.sender,_amountOut) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1019)
Reentrancy in Vault.swap(address,bool,address,address,uint256,bool,uint256,uint64) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#969-1032):
    External calls:
    - _transferInternalBalance(tokenIn,msg.sender,poolAddress,_amountIn) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1010)
        - IMosaicOracle(mosaicOracle).consultUsdPoolPricePerLp(poolAddr,ts) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#703)
        - IFees(fees).onBeforeTransfer(_pool,_from,_to,_internalBalance[_from][_pool],_internalBalance[_to]
[_pool],amount,originalTotalsupplyBase,trader,payload) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#598-601)
        - IPool(token).emitTransfer(from,to,amount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#499)
        - performanceFeesToMint = (IPool(_pool).updatePerfFeePricePerLp(tsTmp,valuePerLp)) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#568)
        - IPool(_pool).emitTransfer(address(0),fees,amount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#610)
        - IFees(fees).onBeforeTransfer(_pool,_from,_to,_internalBalance[_from][_pool],_internalBalance[_to][_pool],amount,0,address(0),payload)
(contracts/mosaic-alpha-contracts/Vault/Vault.sol#614)
    - _transferInternalBalance(tokenOut,poolAddress,msg.sender,_amountOut) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1012)
        - IMosaicOracle(mosaicOracle).consultUsdPoolPricePerLp(poolAddr,ts) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#703)
        - IFees(fees).onBeforeTransfer(_pool,_from,_to,_internalBalance[_from][_pool],_internalBalance[_to]
[_pool],amount,originalTotalsupplyBase,trader,payload) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#598-601)
        - IPool(token).emitTransfer(from,to,amount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#499)
        - performanceFeesToMint = (IPool(_pool).updatePerfFeePricePerLp(tsTmp,valuePerLp)) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#568)
        - IPool(_pool).emitTransfer(address(0),fees,amount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#610)
        - IFees(fees).onBeforeTransfer(_pool,_from,_to,_internalBalance[_from][_pool],_internalBalance[_to][_pool],amount,0,address(0),payload)
(contracts/mosaic-alpha-contracts/Vault/Vault.sol#614)
    - _depositToInternalBalance(tokenIn,msg.sender,poolAddress,_amountIn,true) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1017)
        - returndata = address(token).functionCall(data,SafeERC20: low-level call failed)
(node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#110)
        - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#137)
        - IERC20(token).safeTransferFrom(from,address(this),amount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#463)
    - _withdrawFromInternalBalance(tokenOut,poolAddress,msg.sender,_amountOut) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1019)
        - returndata = address(token).functionCall(data,SafeERC20: low-level call failed)
(node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#110)
        - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#137)
        - IERC20(token).safeTransfer(to,amount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#476)
    - _transferInternalBalance(tokenOut,poolAddress,vaultState.feeTo,_feeToProtocol) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1027)
        - IMosaicOracle(mosaicOracle).consultUsdPoolPricePerLp(poolAddr,ts) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#703)
        - IFees(fees).onBeforeTransfer(_pool,_from,_to,_internalBalance[_from][_pool],_internalBalance[_to]
[_pool],amount,originalTotalsupplyBase,trader,payload) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#598-601)
        - IPool(token).emitTransfer(from,to,amount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#499)
        - performanceFeesToMint = (IPool(_pool).updatePerfFeePricePerLp(tsTmp,valuePerLp)) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#568)
        - IPool(_pool).emitTransfer(address(0),fees,amount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#610)
        - IFees(fees).onBeforeTransfer(_pool,_from,_to,_internalBalance[_from][_pool],_internalBalance[_to][_pool],amount,0,address(0),payload)
(contracts/mosaic-alpha-contracts/Vault/Vault.sol#614)
    - _transferInternalBalance(tokenIn,poolAddress,vaultState.feeTo,_feeToProtocol) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1027)
        - IMosaicOracle(mosaicOracle).consultUsdPoolPricePerLp(poolAddr,ts) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#703)
        - IFees(fees).onBeforeTransfer(_pool,_from,_to,_internalBalance[_from][_pool],_internalBalance[_to]
[_pool],amount,originalTotalsupplyBase,trader,payload) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#598-601)
        - IPool(token).emitTransfer(from,to,amount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#499)
        - performanceFeesToMint = (IPool(_pool).updatePerfFeePricePerLp(tsTmp,valuePerLp)) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#568)
        - IPool(_pool).emitTransfer(address(0),fees,amount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#610)
        - IFees(fees).onBeforeTransfer(_pool,_from,_to,_internalBalance[_from][_pool],_internalBalance[_to][_pool],amount,0,address(0),payload)
(contracts/mosaic-alpha-contracts/Vault/Vault.sol#614)
    External calls sending eth:
    - _depositToInternalBalance(tokenIn,msg.sender,poolAddress,_amountIn,true) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1017)
        - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#137)
    - _withdrawFromInternalBalance(tokenOut,poolAddress,msg.sender,_amountOut) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1019)
        - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#137)
    Event emitted after the call(s):
    - Swap(poolId,tokenIn,tokenOut,_amountIn,_amountOut,msg.sender) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1030)
Reentrancy in Vault.swap(address,bool,address,address,uint256,bool,uint256,uint64) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#969-1032):
    External calls:
    - _transferInternalBalance(tokenIn,msg.sender,poolAddress,_amountIn) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1010)
        - IMosaicOracle(mosaicOracle).consultUsdPoolPricePerLp(poolAddr,ts) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#703)
        - IFees(fees).onBeforeTransfer(_pool,_from,_to,_internalBalance[_from][_pool],_internalBalance[_to]
[_pool],amount,originalTotalsupplyBase,trader,payload) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#598-601)
        - IPool(token).emitTransfer(from,to,amount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#499)
        - performanceFeesToMint = (IPool(_pool).updatePerfFeePricePerLp(tsTmp,valuePerLp)) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#568)
        - IPool(_pool).emitTransfer(address(0),fees,amount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#610)
```

```
        - IFees(fees).onBeforeTransfer(_pool,_from,_to,_internalBalance[_from][_pool],_internalBalance[_to][_pool],amount,0,address(0),payload)
(contracts/mosaic-alpha-contracts/Vault/Vault.sol#614)
    - _transferInternalBalance(tokenOut,poolAddress,msg.sender,_amountOut) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1012)
        - IMosaicOracle(mosaicOracle).consultUsdPoolPricePerLp(poolAddr,ts) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#703)
        - IFees(fees).onBeforeTransfer(_pool,_from,_to,_internalBalance[_from][_pool],_internalBalance[_to]
[_pool],amount,originalTotalsupplyBase,trader,payload) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#598-601)
        - IPool(token).emitTransfer(from,to,amount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#499)
        - performanceFeesToMint = (IPool(_pool).updatePerfFeePricePerLp(tsTmp,valuePerLp)) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#568)
        - IPool(_pool).emitTransfer(address(0),fees,amount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#610)
        - IFees(fees).onBeforeTransfer(_pool,_from,_to,_internalBalance[_from][_pool],_internalBalance[_to][_pool],amount,0,address(0),payload)
(contracts/mosaic-alpha-contracts/Vault/Vault.sol#614)
    - _depositToInternalBalance(tokenIn,msg.sender,poolAddress,_amountIn,true) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1017)
        - returndata = address(token).functionCall(data,SafeERC20: low-level call failed)
(node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#110)
        - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#137)
        - IERC20(token).safeTransferFrom(from,address(this),amount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#463)
    - _withdrawFromInternalBalance(tokenOut,poolAddress,msg.sender,_amountOut) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1019)
        - returndata = address(token).functionCall(data,SafeERC20: low-level call failed)
(node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#110)
        - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#137)
        - IERC20(token).safeTransfer(to,amount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#476)
    - _transferInternalBalance(tokenOut,poolAddress,vaultState.feeTo,_feeToProtocol) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1027)
        - IMosaicOracle(mosaicOracle).consultUsdPoolPricePerLp(poolAddr,ts) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#703)
        - IFees(fees).onBeforeTransfer(_pool,_from,_to,_internalBalance[_from][_pool],_internalBalance[_to]
[_pool],amount,originalTotalsupplyBase,trader,payload) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#598-601)
        - IPool(token).emitTransfer(from,to,amount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#499)
        - performanceFeesToMint = (IPool(_pool).updatePerfFeePricePerLp(tsTmp,valuePerLp)) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#568)
        - IPool(_pool).emitTransfer(address(0),fees,amount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#610)
        - IFees(fees).onBeforeTransfer(_pool,_from,_to,_internalBalance[_from][_pool],_internalBalance[_to][_pool],amount,0,address(0),payload)
(contracts/mosaic-alpha-contracts/Vault/Vault.sol#614)
    External calls sending eth:
    - _depositToInternalBalance(tokenIn,msg.sender,poolAddress,_amountIn,true) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1017)
        - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#137)
    - _withdrawFromInternalBalance(tokenOut,poolAddress,msg.sender,_amountOut) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1019)
        - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#137)
    Event emitted after the call(s):
    - InternalBalanceChanged(from,token,- int256(amount)) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#489)
        - _transferInternalBalance(tokenOut,poolAddress,vaultState.feeTo,_feeToProtocol) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1027)
    - InternalBalanceChanged(to,token,int256(amount)) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#490)
        - _transferInternalBalance(tokenOut,poolAddress,vaultState.feeTo,_feeToProtocol) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1027)
    - InternalBalanceChanged(fees,_pool,int256(uint256(expAmount))) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#609)
        - _transferInternalBalance(tokenOut,poolAddress,vaultState.feeTo,_feeToProtocol) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1027)
    - TotalSupplyBaseChanged(_pool,poolStates[_pool].totalSupplyBase) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#605)
        - _transferInternalBalance(tokenOut,poolAddress,vaultState.feeTo,_feeToProtocol) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1027)
Reentrancy in Vault.swap(address,bool,address,address,uint256,bool,uint256,uint64) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#969-1032):
    External calls:
    - _transferInternalBalance(tokenIn,msg.sender,poolAddress,_amountIn) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1010)
        - IMosaicOracle(mosaicOracle).consultUsdPoolPricePerLp(poolAddr,ts) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#703)
        - IFees(fees).onBeforeTransfer(_pool,_from,_to,_internalBalance[_from][_pool],_internalBalance[_to]
[_pool],amount,originalTotalsupplyBase,trader,payload) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#598-601)
        - IPool(token).emitTransfer(from,to,amount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#499)
        - performanceFeesToMint = (IPool(_pool).updatePerfFeePricePerLp(tsTmp,valuePerLp)) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#568)
        - IPool(_pool).emitTransfer(address(0),fees,amount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#610)
        - IFees(fees).onBeforeTransfer(_pool,_from,_to,_internalBalance[_from][_pool],_internalBalance[_to][_pool],amount,0,address(0),payload)
(contracts/mosaic-alpha-contracts/Vault/Vault.sol#614)
    - _transferInternalBalance(tokenOut,poolAddress,msg.sender,_amountOut) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1012)
        - IMosaicOracle(mosaicOracle).consultUsdPoolPricePerLp(poolAddr,ts) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#703)
        - IFees(fees).onBeforeTransfer(_pool,_from,_to,_internalBalance[_from][_pool],_internalBalance[_to]
[_pool],amount,originalTotalsupplyBase,trader,payload) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#598-601)
        - IPool(token).emitTransfer(from,to,amount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#499)
        - performanceFeesToMint = (IPool(_pool).updatePerfFeePricePerLp(tsTmp,valuePerLp)) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#568)
        - IPool(_pool).emitTransfer(address(0),fees,amount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#610)
        - IFees(fees).onBeforeTransfer(_pool,_from,_to,_internalBalance[_from][_pool],_internalBalance[_to][_pool],amount,0,address(0),payload)
(contracts/mosaic-alpha-contracts/Vault/Vault.sol#614)
    - _depositToInternalBalance(tokenIn,msg.sender,poolAddress,_amountIn,true) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1017)
        - returndata = address(token).functionCall(data,SafeERC20: low-level call failed)
(node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#110)
        - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#137)
        - IERC20(token).safeTransferFrom(from,address(this),amount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#463)
    - _withdrawFromInternalBalance(tokenOut,poolAddress,msg.sender,_amountOut) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1019)
        - returndata = address(token).functionCall(data,SafeERC20: low-level call failed)
(node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#110)
        - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#137)
        - IERC20(token).safeTransfer(to,amount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#476)
    - _transferInternalBalance(tokenIn,poolAddress,vaultState.feeTo,_feeToProtocol) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1027)
        - IMosaicOracle(mosaicOracle).consultUsdPoolPricePerLp(poolAddr,ts) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#703)
        - IFees(fees).onBeforeTransfer(_pool,_from,_to,_internalBalance[_from][_pool],_internalBalance[_to]
[_pool],amount,originalTotalsupplyBase,trader,payload) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#598-601)
        - IPool(token).emitTransfer(from,to,amount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#499)
        - performanceFeesToMint = (IPool(_pool).updatePerfFeePricePerLp(tsTmp,valuePerLp)) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#568)
        - IPool(_pool).emitTransfer(address(0),fees,amount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#610)
        - IFees(fees).onBeforeTransfer(_pool,_from,_to,_internalBalance[_from][_pool],_internalBalance[_to][_pool],amount,0,address(0),payload)
(contracts/mosaic-alpha-contracts/Vault/Vault.sol#614)
    External calls sending eth:
    - _depositToInternalBalance(tokenIn,msg.sender,poolAddress,_amountIn,true) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1017)
        - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#137)
    - _withdrawFromInternalBalance(tokenOut,poolAddress,msg.sender,_amountOut) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1019)
        - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#137)
    Event emitted after the call(s):
    - InternalBalanceChanged(from,token,- int256(amount)) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#489)
        - _transferInternalBalance(tokenIn,poolAddress,vaultState.feeTo,_feeToProtocol) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1027)
    - InternalBalanceChanged(to,token,int256(amount)) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#490)
        - _transferInternalBalance(tokenIn,poolAddress,vaultState.feeTo,_feeToProtocol) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1027)
    - InternalBalanceChanged(fees,_pool,int256(uint256(expAmount))) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#609)
        - _transferInternalBalance(tokenIn,poolAddress,vaultState.feeTo,_feeToProtocol) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1027)
    - TotalSupplyBaseChanged(_pool,poolStates[_pool].totalSupplyBase) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#605)
        - _transferInternalBalance(tokenIn,poolAddress,vaultState.feeTo,_feeToProtocol) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1027)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3

Reentrancy in Fees._selfManageMeBefore() (contracts/mosaic-alpha-contracts/Fees/Fees.sol#145-161):
    External calls:
    - IUserProfile(userProfile).registerUser(slots[1]) (contracts/mosaic-alpha-contracts/Fees/Fees.sol#148)
    Event emitted after the call(s):
    - OwnershipTransferred(oldOwner,newOwner) (contracts/mosaic-alpha-contracts/helpers/Ownable.sol#79)
        - _transferOwnership(slots[0]) (contracts/mosaic-alpha-contracts/Fees/Fees.sol#152)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3

Reentrancy in AffiliateBooster._allocateTicket(address,address,uint256,uint256) (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#352-365):
    External calls:
    - userProfile.registerUser(_a,_referredBy) (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#353)
    - controller.ticketAllocated(_a,_c) (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#362)
    Event emitted after the call(s):
    - TicketBought(state.currentEpoch,_a,_c,_price) (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#364)
Reentrancy in SimpleDeposit._deposit(address,address,uint256) (contracts/mosaic-alpha-contracts/Deposit/SimpleDeposit.sol#269-293):
    External calls:
    - successTransfer = _stakingToken.transferFrom(_sender,address(this),_amount) (contracts/mosaic-alpha-contracts/Deposit/SimpleDeposit.sol#279-283)
    Event emitted after the call(s):
```

```
    - DepositCreated(_user,_amount,block.timestamp,totalDeposits,accountBalances[_user].balance) (contracts/mosaic-alpha-
contracts/Deposit/SimpleDeposit.sol#292)
Reentrancy in UserProfile._registerUser(address,address) (contracts/mosaic-alpha-contracts/User/UserProfile.sol#226-247):
    External calls:
    - _referredBy = IAffiliateBooster(affiliateBoosterContract).pickNextReferral() (contracts/mosaic-alpha-contracts/User/UserProfile.sol#230)
    - _registerUser(_referredBy,address(0)) (contracts/mosaic-alpha-contracts/User/UserProfile.sol#236)
        - _referredBy = IAffiliateBooster(affiliateBoosterContract).pickNextReferral() (contracts/mosaic-alpha-contracts/User/UserProfile.sol#230)
    Event emitted after the call(s):
    - UserRegistered(_user,_referredBy,users[_user].referredByRank,users[_user].buyFeeDiscount) (contracts/mosaic-alpha-contracts/User/UserProfile.sol#246)
    - UserRegistered(_user,_referredBy,users[_user].referredByRank,users[_user].buyFeeDiscount) (contracts/mosaic-alpha-contracts/User/UserProfile.sol#246)
        - _registerUser(_referredBy,address(0)) (contracts/mosaic-alpha-contracts/User/UserProfile.sol#236)
Reentrancy in SimpleDeposit._userLock(address,uint256,address) (contracts/mosaic-alpha-contracts/Deposit/SimpleDeposit.sol#320-336):
    External calls:
    - _handleUserStakeChanged(_user,_referredBy,accountBalances[_user].lockedBalance) (contracts/mosaic-alpha-contracts/Deposit/SimpleDeposit.sol#326)
        - IAffiliate(_affiliateContract).userStakeChanged(_user,_referredBy,_amount) (contracts/mosaic-alpha-contracts/Deposit/SimpleDeposit.sol#130)
        - IFees(_feesContract).userStakeChanged(_user,_amount) (contracts/mosaic-alpha-contracts/Deposit/SimpleDeposit.sol#133)
    Event emitted after the call(s):
    - DepositLocked(_user,_amount,block.timestamp,block.timestamp + lockTime,totalLocked,accountBalances[_user].lockedBalance) (contracts/mosaic-alpha-
contracts/Deposit/SimpleDeposit.sol#328-335)
Reentrancy in SimpleDeposit._userUnlock(address,uint256) (contracts/mosaic-alpha-contracts/Deposit/SimpleDeposit.sol#338-355):
    External calls:
    - _handleUserStakeChanged(_account,address(0),accountBalances[_account].lockedBalance) (contracts/mosaic-alpha-contracts/Deposit/SimpleDeposit.sol#346)
        - IAffiliate(_affiliateContract).userStakeChanged(_user,_referredBy,_amount) (contracts/mosaic-alpha-contracts/Deposit/SimpleDeposit.sol#130)
        - IFees(_feesContract).userStakeChanged(_user,_amount) (contracts/mosaic-alpha-contracts/Deposit/SimpleDeposit.sol#133)
    Event emitted after the call(s):
    - DepositUnlocked(msg.sender,_amount,block.timestamp,totalLocked,accountBalances[_account].lockedBalance) (contracts/mosaic-alpha-
contracts/Deposit/SimpleDeposit.sol#348-354)
Reentrancy in SimpleDeposit._withdraw(address,uint256) (contracts/mosaic-alpha-contracts/Deposit/SimpleDeposit.sol#295-318):
    External calls:
    - successTransfer = _stakingToken.transfer(_user,_amount) (contracts/mosaic-alpha-contracts/Deposit/SimpleDeposit.sol#306)
    Event emitted after the call(s):
    - DepositWithdrawn(_user,_amount,block.timestamp,totalDeposits,accountBalances[_user].balance) (contracts/mosaic-alpha-
contracts/Deposit/SimpleDeposit.sol#311-317)
Reentrancy in Swaps.addTokens(address[]) (contracts/mosaic-alpha-contracts/Vault/Swaps.sol#29-38):
    External calls:
    - _approveToken(token,type()(uint256).max) (contracts/mosaic-alpha-contracts/Vault/Swaps.sol#33)
        - IERC20(_token).approve(vaultAddress,_amount) (contracts/mosaic-alpha-contracts/Vault/Swaps.sol#25)
        - IERC20(_token).approve(pancakeRouterAddress,_amount) (contracts/mosaic-alpha-contracts/Vault/Swaps.sol#26)
    Event emitted after the call(s):
    - tokenAdded(token) (contracts/mosaic-alpha-contracts/Vault/Swaps.sol#35)
Reentrancy in AffiliateBooster.allocateTicket(address,address,uint256,uint256) (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#340-343):
    External calls:
    - _allocateTicket(_a,_referredBy,_c,_price) (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#342)
        - userProfile.registerUser(_a,_referredBy) (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#353)
        - controller.ticketAllocated(_a,_c) (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#362)
    - closeEpochIfTimeUp() (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#340)
        - controller.epochClosed(state.currentEpoch - 1) (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#427)
    Event emitted after the call(s):
    - TicketBought(state.currentEpoch,_a,_c,_price) (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#364)
        - _allocateTicket(_a,_referredBy,_c,_price) (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#342)
Reentrancy in SimpleDeposit.depositAndLock(uint256) (contracts/mosaic-alpha-contracts/Deposit/SimpleDeposit.sol#200-203):
    External calls:
    - _deposit(msg.sender,msg.sender,_amount) (contracts/mosaic-alpha-contracts/Deposit/SimpleDeposit.sol#201)
        - successTransfer = _stakingToken.transferFrom(_sender,address(this),_amount) (contracts/mosaic-alpha-contracts/Deposit/SimpleDeposit.sol#279-283)
    - _userLock(msg.sender,_amount,address(0)) (contracts/mosaic-alpha-contracts/Deposit/SimpleDeposit.sol#202)
        - IAffiliate(_affiliateContract).userStakeChanged(_user,_referredBy,_amount) (contracts/mosaic-alpha-contracts/Deposit/SimpleDeposit.sol#130)
        - IFees(_feesContract).userStakeChanged(_user,_amount) (contracts/mosaic-alpha-contracts/Deposit/SimpleDeposit.sol#133)
    Event emitted after the call(s):
    - DepositLocked(_user,_amount,block.timestamp,block.timestamp + lockTime,totalLocked,accountBalances[_user].lockedBalance) (contracts/mosaic-alpha-
contracts/Deposit/SimpleDeposit.sol#328-335)
        - _userLock(msg.sender,_amount,address(0)) (contracts/mosaic-alpha-contracts/Deposit/SimpleDeposit.sol#202)
Reentrancy in SimpleDeposit.depositAndLock(uint256,address) (contracts/mosaic-alpha-contracts/Deposit/SimpleDeposit.sol#205-208):
    External calls:
    - _deposit(msg.sender,msg.sender,_amount) (contracts/mosaic-alpha-contracts/Deposit/SimpleDeposit.sol#206)
        - successTransfer = _stakingToken.transferFrom(_sender,address(this),_amount) (contracts/mosaic-alpha-contracts/Deposit/SimpleDeposit.sol#279-283)
    - _userLock(msg.sender,_amount,_referredBy) (contracts/mosaic-alpha-contracts/Deposit/SimpleDeposit.sol#207)
        - IAffiliate(_affiliateContract).userStakeChanged(_user,_referredBy,_amount) (contracts/mosaic-alpha-contracts/Deposit/SimpleDeposit.sol#130)
        - IFees(_feesContract).userStakeChanged(_user,_amount) (contracts/mosaic-alpha-contracts/Deposit/SimpleDeposit.sol#133)
    Event emitted after the call(s):
    - DepositLocked(_user,_amount,block.timestamp,block.timestamp + lockTime,totalLocked,accountBalances[_user].lockedBalance) (contracts/mosaic-alpha-
contracts/Deposit/SimpleDeposit.sol#328-335)
        - _userLock(msg.sender,_amount,_referredBy) (contracts/mosaic-alpha-contracts/Deposit/SimpleDeposit.sol#207)
Reentrancy in AffiliateBooster.pickNextReferral() (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#372-410):
    External calls:
    - controller.referralPicked(address(0),true) (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#375)
    - closeEpochIfTimeUp() (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#372)
        - controller.epochClosed(state.currentEpoch - 1) (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#427)
    Event emitted after the call(s):
    - ReferralPicked(address(0)) (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#376)
Reentrancy in AffiliateBooster.pickNextReferral() (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#372-410):
    External calls:
    - closeEpochIfTimeUp() (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#372)
        - controller.epochClosed(state.currentEpoch - 1) (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#427)
    Event emitted after the call(s):
    - TicketSpent(state.currentEpoch,pickedAddress) (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#401)
Reentrancy in AffiliateBooster.pickNextReferral() (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#372-410):
    External calls:
    - controller.referralPicked(pickedAddress,false) (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#404)
    - _incrementEpoch() (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#407)
        - controller.epochClosed(state.currentEpoch - 1) (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#427)
    - closeEpochIfTimeUp() (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#372)
        - controller.epochClosed(state.currentEpoch - 1) (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#427)
    Event emitted after the call(s):
    - EpochClosed(state.currentEpoch - 1) (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#425)
        - _incrementEpoch() (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#407)
    - ReferralPicked(pickedAddress) (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#409)
Reentrancy in AffiliateBooster.purchaseTicket(uint256) (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#313-315):
    External calls:
    - purchaseTicket(address(0),_c) (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#314)
        - userProfile.registerUser(_a,_referredBy) (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#353)
        - require(bool,string)(IERC20Burnable(state.feeToken).transferFrom(msg.sender,address(this),_c * state.ticketPrice),TransferFrom failed)
(contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#324)
        - IERC20Burnable(state.feeToken).burn(_c * state.ticketPrice) (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#326)
        - IERC20Burnable(state.feeToken).transfer(state.feeTo,_c * state.ticketPrice) (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#328)
        - controller.epochClosed(state.currentEpoch - 1) (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#427)
        - controller.ticketAllocated(_a,_c) (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#362)
    - closeEpochIfTimeUp() (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#313)
        - controller.epochClosed(state.currentEpoch - 1) (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#427)
    Event emitted after the call(s):
    - EpochClosed(state.currentEpoch - 1) (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#425)
        - purchaseTicket(address(0),_c) (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#314)
    - TicketBought(state.currentEpoch,_a,_c,_price) (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#364)
        - purchaseTicket(address(0),_c) (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#314)
Reentrancy in AffiliateBooster.purchaseTicket(address,uint256) (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#322-331):
    External calls:
    - require(bool,string)(IERC20Burnable(state.feeToken).transferFrom(msg.sender,address(this),_c * state.ticketPrice),TransferFrom failed)
(contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#324)
```

```
    - IERC20Burnable(state.feeToken).burn(_c * state.ticketPrice) (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#326)
    - IERC20Burnable(state.feeToken).transfer(state.feeTo,_c * state.ticketPrice) (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#328)
    - _allocateTicket(msg.sender,_referredBy,_c,state.ticketPrice) (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#330)
        - userProfile.registerUser(_a,_referredBy) (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#353)
        - controller.ticketAllocated(_a,_c) (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#362)
    - closeEpochIfTimeUp() (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#322)
        - controller.epochClosed(state.currentEpoch - 1) (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#427)
    Event emitted after the call(s):
    - TicketBought(state.currentEpoch,_a,_c,_price) (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#364)
        - _allocateTicket(msg.sender,_referredBy,_c,state.ticketPrice) (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#330)
Reentrancy in Swaps.removeTokens(address[]) (contracts/mosaic-alpha-contracts/Vault/Swaps.sol#40-49):
    External calls:
    - _approveToken(token,0) (contracts/mosaic-alpha-contracts/Vault/Swaps.sol#44)
        - IERC20(_token).approve(vaultAddress,_amount) (contracts/mosaic-alpha-contracts/Vault/Swaps.sol#25)
        - IERC20(_token).approve(pancakeRouterAddress,_amount) (contracts/mosaic-alpha-contracts/Vault/Swaps.sol#26)
    Event emitted after the call(s):
    - tokenRemoved(token) (contracts/mosaic-alpha-contracts/Vault/Swaps.sol#46)
Reentrancy in AffiliateBooster.setEpochWeights(uint8[]) (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#287-291):
    External calls:
    - _incrementEpoch() (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#289)
        - controller.epochClosed(state.currentEpoch - 1) (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#427)
    Event emitted after the call(s):
    - EpochWeightsChanged(_weights) (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#290)
Reentrancy in SimpleDeposit.unlockAndWithdraw(uint256) (contracts/mosaic-alpha-contracts/Deposit/SimpleDeposit.sol#210-213):
    External calls:
    - _userUnlock(msg.sender,_amount) (contracts/mosaic-alpha-contracts/Deposit/SimpleDeposit.sol#211)
        - IAffiliate(_affiliateContract).userStakeChanged(_user,_referredBy,_amount) (contracts/mosaic-alpha-contracts/Deposit/SimpleDeposit.sol#130)
        - IFees(_feesContract).userStakeChanged(_user,_amount) (contracts/mosaic-alpha-contracts/Deposit/SimpleDeposit.sol#133)
    - _withdraw(msg.sender,_amount) (contracts/mosaic-alpha-contracts/Deposit/SimpleDeposit.sol#212)
        - successTransfer = _stakingToken.transfer(_user,_amount) (contracts/mosaic-alpha-contracts/Deposit/SimpleDeposit.sol#306)
    Event emitted after the call(s):
    - DepositWithdrawn(_user,_amount,block.timestamp,totalDeposits,accountBalances[_user].balance) (contracts/mosaic-alpha-
contracts/Deposit/SimpleDeposit.sol#311-317)
        - _withdraw(msg.sender,_amount) (contracts/mosaic-alpha-contracts/Deposit/SimpleDeposit.sol#212)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
```

block-timestamp

```
PancakeERC20.permit(address,address,uint256,uint256,uint8,bytes32,bytes32) (contracts/mosaic-alpha-contracts/helpers/pancake/PancakeERC20.sol#83-95) uses
timestamp for comparisons
    Dangerous comparisons:
    - require(bool,string)(deadline >= block.timestamp,Pancake: EXPIRED) (contracts/mosaic-alpha-contracts/helpers/pancake/PancakeERC20.sol#84)
PancakePair._update(uint256,uint256,uint112,uint112) (contracts/mosaic-alpha-contracts/helpers/pancake/PancakePair.sol#75-88) uses timestamp for
comparisons
    Dangerous comparisons:
    - timeElapsed > 0 && _reserve0 != 0 && _reserve1 != 0 (contracts/mosaic-alpha-contracts/helpers/pancake/PancakePair.sol#79)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp

Pool.removeToken(uint256) (contracts/mosaic-alpha-contracts/Pool/Pool.sol#569-590) uses timestamp for comparisons
    Dangerous comparisons:
    - poolType == 0 && weightChangeEnd >= block.timestamp (contracts/mosaic-alpha-contracts/Pool/Pool.sol#574)
Pool.calcWeight(uint32,uint32,uint32,uint32) (contracts/mosaic-alpha-contracts/Pool/Pool.sol#685-710) uses timestamp for comparisons
    Dangerous comparisons:
    - _now >= _changeEnd (contracts/mosaic-alpha-contracts/Pool/Pool.sol#687)
    - _now <= _changeStart (contracts/mosaic-alpha-contracts/Pool/Pool.sol#691)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp

Governance.transfer_proportion(address,uint256) (contracts/mosaic-alpha-contracts/Governance/Governance.sol#174-182) uses timestamp for comparisons
    Dangerous comparisons:
    - require(bool,string)(block.timestamp >= action_curator_timer[msg.sender] + vote_time_threshold,Not yet, your votes need to conclude)
(contracts/mosaic-alpha-contracts/Governance/Governance.sol#176)
Governance.propose_config(string,bool,address,address[],bool[],address[],uint256[],bytes32[]) (contracts/mosaic-alpha-
contracts/Governance/Governance.sol#229-260) uses timestamp for comparisons
    Dangerous comparisons:
    - require(bool,string)(block.timestamp >= conf_curator_timer[msg.sender] + vote_conf_time_threshold,Curator timer not yet expired) (contracts/mosaic-
alpha-contracts/Governance/Governance.sol#240)
    - require(bool,string)(conf_time_limit[conf_counter] == 0,In progress) (contracts/mosaic-alpha-contracts/Governance/Governance.sol#242)
Governance.propose_core_change(address,bool,address[],address[]) (contracts/mosaic-alpha-contracts/Governance/Governance.sol#263-279) uses timestamp for
comparisons
    Dangerous comparisons:
    - require(bool,string)(block.timestamp >= conf_curator_timer[msg.sender] + vote_conf_time_threshold,Curator timer not yet expired) (contracts/mosaic-
alpha-contracts/Governance/Governance.sol#265)
    - require(bool,string)(conf_time_limit[conf_counter] == 0,In progress) (contracts/mosaic-alpha-contracts/Governance/Governance.sol#266)
Governance.support_config_proposal(uint256,string) (contracts/mosaic-alpha-contracts/Governance/Governance.sol#282-317) uses timestamp for comparisons
    Dangerous comparisons:
    - require(bool,string)(block.timestamp >= conf_curator_timer[msg.sender] + vote_conf_time_threshold,Curator timer not yet expired) (contracts/mosaic-
alpha-contracts/Governance/Governance.sol#284)
    - require(bool,string)(conf_time_limit[_confCount] > block.timestamp,Timed out) (contracts/mosaic-alpha-contracts/Governance/Governance.sol#285)
    - require(bool,string)(conf_time_limit[_confCount] != 0,Not started) (contracts/mosaic-alpha-contracts/Governance/Governance.sol#286)
Governance.propose_action(uint256,address,bytes) (contracts/mosaic-alpha-contracts/Governance/Governance.sol#417-430) uses timestamp for comparisons
    Dangerous comparisons:
    - require(bool,string)(action_time_limit[_id] == 0,Create a new one) (contracts/mosaic-alpha-contracts/Governance/Governance.sol#420)
    - require(bool,string)(block.timestamp >= action_curator_timer[msg.sender] + vote_time_threshold,Not yet) (contracts/mosaic-alpha-
contracts/Governance/Governance.sol#421)
Governance.support_actions(uint256) (contracts/mosaic-alpha-contracts/Governance/Governance.sol#433-441) uses timestamp for comparisons
    Dangerous comparisons:
    - require(bool,string)(block.timestamp >= action_curator_timer[msg.sender] + vote_time_threshold,Not yet) (contracts/mosaic-alpha-
contracts/Governance/Governance.sol#435)
    - require(bool,string)(action_time_limit[_id] > block.timestamp,Action timed out) (contracts/mosaic-alpha-contracts/Governance/Governance.sol#436)
Governance.trigger_action(uint256) (contracts/mosaic-alpha-contracts/Governance/Governance.sol#445-454) uses timestamp for comparisons
    Dangerous comparisons:
    - require(bool,string)(action_time_limit[_id] > block.timestamp,Action timed out) (contracts/mosaic-alpha-contracts/Governance/Governance.sol#447)
Vault.LiveFun(address) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#141-143) uses timestamp for comparisons
    Dangerous comparisons:
    - vaultState.emergencyMode || poolStates[_a].poolEmergencyMode (contracts/mosaic-alpha-contracts/Vault/Vault.sol#142)
Vault._calculateTrailingAndPerformanceFee(address,bool,bool) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#544-572) uses timestamp for comparisons
    Dangerous comparisons:
    - (forcePerformance && (poolStates[_pool].lastPerformanceTimestamp < block.timestamp)) || poolStates[_pool].lastPerformanceTimestamp < block.timestamp
- 604800 (contracts/mosaic-alpha-contracts/Vault/Vault.sol#548)
    - (forceTrailing && (poolStates[_pool].lastTrailingTimestamp < block.timestamp)) || poolStates[_pool].lastTrailingTimestamp < block.timestamp - 259200
(contracts/mosaic-alpha-contracts/Vault/Vault.sol#554)
Vault.registerTokens(address[],bool) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#743-764) uses timestamp for comparisons
    Dangerous comparisons:
    - poolStates[msg.sender].poolTokenCount > vaultState.maxPoolTokenCount (contracts/mosaic-alpha-contracts/Vault/Vault.sol#761)
Vault.Mint(uint32,uint256,uint256[],address,address,bool,uint256,uint256) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#841-904) uses timestamp for
comparisons
    Dangerous comparisons:
    - block.timestamp > deadline (contracts/mosaic-alpha-contracts/Vault/Vault.sol#842)
Vault.Burn(uint32,uint256,uint256[],bool,uint256,address) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#913-951) uses timestamp for comparisons
    Dangerous comparisons:
    - block.timestamp > deadline (contracts/mosaic-alpha-contracts/Vault/Vault.sol#914)
Vault.swap(address,bool,address,address,uint256,bool,uint256,uint64) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#969-1032) uses timestamp for
comparisons
    Dangerous comparisons:
    - block.timestamp > deadline (contracts/mosaic-alpha-contracts/Vault/Vault.sol#974)
Vault.isRunning(uint256) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1059-1065) uses timestamp for comparisons
```

```
    Dangerous comparisons:
    - auctions[auctionId].remainingAmount == 0 || _expiryTime < block.timestamp (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1061)
Vault.startAuction(address,uint256,address,uint32,uint32,uint256) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1071-1108) uses timestamp for
comparisons
    Dangerous comparisons:
    - duration < vaultState.bidMinDuration (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1078)
    - endingPrice > startingPrice (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1084)
Vault._getBidPrice(uint256,uint32) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1130-1151) uses timestamp for comparisons
    Dangerous comparisons:
    - _elapsed > _duration (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1138)
    - _now >= _end || remainingAmount == 0 (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1141)
Vault.bid(uint256,uint256,bool,bool,uint256) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1163-1213) uses timestamp for comparisons
    Dangerous comparisons:
    - block.timestamp > deadline (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1164)
    - _auctionDetails.startsAt + 60 >= block.timestamp (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1177)
Vault.flashLoan(IERC3156FlashBorrower,address,uint256,bytes) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1262-1289) uses timestamp for comparisons
    Dangerous comparisons:
    - receiver.onFlashLoan(msg.sender,token,amount,_fee,data) != CALLBACK_SUCCESS (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1280)
    - IERC20(token).balanceOf(address(this)) != lastBalance + amount + _fee (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1286)
Vault.emergencyWithdraw(address,address,bool) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1330-1360) uses timestamp for comparisons
    Dangerous comparisons:
    - ! poolStates[poolAddress].poolEmergencyMode && ! vaultState.emergencyMode (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1332)
Vault.emergencyWithdrawAll(address,address,uint256) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1364-1368) uses timestamp for comparisons
    Dangerous comparisons:
    - vaultState.emergencyMode != true || block.timestamp <= vaultState.emergencyModeTimestamp + 31536000 (contracts/mosaic-alpha-
contracts/Vault/Vault.sol#1366)
Vault.emergencyWithdrawPool(address,address,address,uint256) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1372-1376) uses timestamp for comparisons
    Dangerous comparisons:
    - poolStates[_pool].poolEmergencyMode != true || block.timestamp <= poolStates[_pool].emergencyModeTime + 31536000 (contracts/mosaic-alpha-
contracts/Vault/Vault.sol#1374)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp

Fees._updateUserTimeIntegrals(address,uint32,uint8,uint32) (contracts/mosaic-alpha-contracts/Fees/Fees.sol#185-197) uses timestamp for comparisons
    Dangerous comparisons:
    - timeDelta == 0 (contracts/mosaic-alpha-contracts/Fees/Fees.sol#187)
Fees.claimUserClaimableFee(address,address) (contracts/mosaic-alpha-contracts/Fees/Fees.sol#205-305) uses timestamp for comparisons
    Dangerous comparisons:
    - timeDelta == 0 (contracts/mosaic-alpha-contracts/Fees/Fees.sol#207)
    - timeDelta == block.timestamp (contracts/mosaic-alpha-contracts/Fees/Fees.sol#245)
    - discountNow > discountLast (contracts/mosaic-alpha-contracts/Fees/Fees.sol#258)
    - l1Now > l1Last (contracts/mosaic-alpha-contracts/Fees/Fees.sol#264)
    - l2Now > l2Last (contracts/mosaic-alpha-contracts/Fees/Fees.sol#270)
    - affiToPay > 0 (contracts/mosaic-alpha-contracts/Fees/Fees.sol#276)
    - toPay > 0 (contracts/mosaic-alpha-contracts/Fees/Fees.sol#302)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp

SimpleDeposit.getWithdrawableAccountLockedBalance(address) (contracts/mosaic-alpha-contracts/Deposit/SimpleDeposit.sol#250-253) uses timestamp for
comparisons
    Dangerous comparisons:
    - accountBalances[_account].lastLockTimestamp + lockTime > block.timestamp (contracts/mosaic-alpha-contracts/Deposit/SimpleDeposit.sol#251)
SimpleDeposit._userUnlock(address,uint256) (contracts/mosaic-alpha-contracts/Deposit/SimpleDeposit.sol#338-355) uses timestamp for comparisons
    Dangerous comparisons:
    - require(bool,string)(accountBalances[_account].lastLockTimestamp + lockTime < uint64(block.timestamp),DEPOSIT LOCKED) (contracts/mosaic-alpha-
contracts/Deposit/SimpleDeposit.sol#339-342)
AffiliateBooster.closeEpoch() (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#108-117) uses timestamp for comparisons
    Dangerous comparisons:
    - state.epochTime > 0 && block.timestamp >= state.currentEpochStart + state.epochTime (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#109)
    - state.epochSize > 0 && epochs[state.currentEpoch].ticketsSpent >= state.epochSize (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#112)
AffiliateBooster.purchaseTicket(address,uint256) (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#322-331) uses timestamp for comparisons
    Dangerous comparisons:
    - require(bool,string)(IERC20Burnable(state.feeToken).transferFrom(msg.sender,address(this),_c * state.ticketPrice),TransferFrom failed)
(contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#324)
AffiliateBooster.pickNextReferral() (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#372-410) uses timestamp for comparisons
    Dangerous comparisons:
    - state.totalPlayerWeight == 0 (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#374)
    - threshold < sumWeight (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#388)
    - state.epochSize > 0 && epochs[state.currentEpoch].ticketsSpent >= state.epochSize (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#406)
AffiliateBooster._incrementEpoch() (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#415-429) uses timestamp for comparisons
    Dangerous comparisons:
    - i < state.epochWeights.length && i <= state.currentEpoch (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#422)
AffiliateBooster.getUserTickets(address,uint256[]) (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#475-480) uses timestamp for comparisons
    Dangerous comparisons:
    - epochIds[i] + state.epochWeights.length > state.currentEpoch (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#478)
PancakeOracleLibrary.currentCumulativePrices(address) (contracts/mosaic-alpha-contracts/helpers/pancake/libraries/PancakeOracleLibrary.sol#16-34) uses
timestamp for comparisons
    Dangerous comparisons:
    - blockTimestampLast != blockTimestamp (contracts/mosaic-alpha-contracts/helpers/pancake/libraries/PancakeOracleLibrary.sol#25)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
```

assembly-usage

```
PancakeERC20.constructor() (contracts/mosaic-alpha-contracts/helpers/pancake/PancakeERC20.sol#26-40) uses assembly
    - INLINE ASM (contracts/mosaic-alpha-contracts/helpers/pancake/PancakeERC20.sol#28-30)
PancakeFactory.createPair(address,address) (contracts/mosaic-alpha-contracts/helpers/pancake/PancakeFactory.sol#29-44) uses assembly
    - INLINE ASM (contracts/mosaic-alpha-contracts/helpers/pancake/PancakeFactory.sol#36-38)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

FixedPoint.mulUp(uint256,uint256) (contracts/mosaic-alpha-contracts/Pool/Math.sol#20-35) uses assembly
    - INLINE ASM (contracts/mosaic-alpha-contracts/Pool/Math.sol#32-34)
FixedPoint.divUp(uint256,uint256) (contracts/mosaic-alpha-contracts/Pool/Math.sol#46-63) uses assembly
    - INLINE ASM (contracts/mosaic-alpha-contracts/Pool/Math.sol#60-62)
FixedPoint.complement(uint256) (contracts/mosaic-alpha-contracts/Pool/Math.sol#119-125) uses assembly
    - INLINE ASM (contracts/mosaic-alpha-contracts/Pool/Math.sol#122-124)
Pool.addToken(address) (contracts/mosaic-alpha-contracts/Pool/Pool.sol#535-559) uses assembly
    - INLINE ASM (contracts/mosaic-alpha-contracts/Pool/Pool.sol#558)
console._sendLogPayload(bytes) (node_modules/hardhat/console.sol#7-14) uses assembly
    - INLINE ASM (node_modules/hardhat/console.sol#10-13)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

Address.verifyCallResult(bool,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#201-221) uses assembly
    - INLINE ASM (node_modules/@openzeppelin/contracts/utils/Address.sol#213-216)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

Address._revert(bytes,string) (contracts/.deps/npm/@openzeppelin/contracts/utils/Address.sol#231-243) uses assembly
    - INLINE ASM (contracts/.deps/npm/@openzeppelin/contracts/utils/Address.sol#236-239)
Initializable.isConstructor() (contracts/mosaic-alpha-contracts/Oracle/Initializable.sol#53-63) uses assembly
    - INLINE ASM (contracts/mosaic-alpha-contracts/Oracle/Initializable.sol#61)
PoolFactory.create(string,string,uint8,address[],uint32[],uint256[],IFees.MosaicPoolFees,address,bool) (contracts/mosaic-alpha-
contracts/Pool/PoolFactory.sol#120-263) uses assembly
    - INLINE ASM (contracts/mosaic-alpha-contracts/Pool/PoolFactory.sol#165-174)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
```

boolean-equality

```
Pool.removeExpired() (contracts/mosaic-alpha-contracts/Pool/Pool.sol#598-608) compares to a boolean constant:
    -IVault(vaultAddress).isRunning(runningDutchAuctionIds[i]) == false (contracts/mosaic-alpha-contracts/Pool/Pool.sol#601)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#boolean-equality

Governance.onlyManagers() (contracts/mosaic-alpha-contracts/Governance/Governance.sol#112-119) compares to a boolean constant:
    -ok == true (contracts/mosaic-alpha-contracts/Governance/Governance.sol#118)
Governance.ActivateDeployedMosaic(address,address,address,address,address,address,address,address,address,address) (contracts/mosaic-alpha-
contracts/Governance/Governance.sol#128-171) compares to a boolean constant:
    -require(bool,string)(deployed == false,It is done.) (contracts/mosaic-alpha-contracts/Governance/Governance.sol#142)
Governance.support_config_proposal(uint256,string) (contracts/mosaic-alpha-contracts/Governance/Governance.sol#282-317) compares to a boolean constant:
    -conf_votes[_confCount] >= action_threshold && triggered[_confCount] == false (contracts/mosaic-alpha-contracts/Governance/Governance.sol#290)
Governance.trigger_action(uint256) (contracts/mosaic-alpha-contracts/Governance/Governance.sol#445-454) compares to a boolean constant:
    -require(bool,string)(triggered[_id] == false,Already triggered) (contracts/mosaic-alpha-contracts/Governance/Governance.sol#449)
Vault.setPoolEmergencyMode(address) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#131-136) compares to a boolean constant:
    -poolStates[poolAddress].poolEmergencyMode != false (contracts/mosaic-alpha-contracts/Vault/Vault.sol#133)
Vault.Mint(uint32,uint256,uint256[],address,address,bool,uint256,uint256) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#841-904) compares to a boolean
constant:
    -vaultState.whitelistingEnabled == true (contracts/mosaic-alpha-contracts/Vault/Vault.sol#847)
Vault.emergencyWithdrawAll(address,address,uint256) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1364-1368) compares to a boolean constant:
    -vaultState.emergencyMode != true || block.timestamp <= vaultState.emergencyModeTimestamp + 31536000 (contracts/mosaic-alpha-
contracts/Vault/Vault.sol#1366)
Vault.emergencyWithdrawPool(address,address,address,uint256) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1372-1376) compares to a boolean constant:
    -poolStates[_pool].poolEmergencyMode != true || block.timestamp <= poolStates[_pool].emergencyModeTime + 31536000 (contracts/mosaic-alpha-
contracts/Vault/Vault.sol#1374)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#boolean-equality

SimpleDeposit.disapproveExternalAddress(address) (contracts/mosaic-alpha-contracts/Deposit/SimpleDeposit.sol#145-150) compares to a boolean constant:
    -approvedExternalAddresses[msg.sender][_locker] == true (contracts/mosaic-alpha-contracts/Deposit/SimpleDeposit.sol#146)
SimpleDeposit._deposit(address,address,uint256) (contracts/mosaic-alpha-contracts/Deposit/SimpleDeposit.sol#269-293) compares to a boolean constant:
    -whitelistingEnabled == true (contracts/mosaic-alpha-contracts/Deposit/SimpleDeposit.sol#272)
SimpleDeposit._deposit(address,address,uint256) (contracts/mosaic-alpha-contracts/Deposit/SimpleDeposit.sol#269-293) compares to a boolean constant:
    -require(bool,string)(_registerContract.isWhitelisted(_user) == true,Caller is not whitelisted!) (contracts/mosaic-alpha-
contracts/Deposit/SimpleDeposit.sol#273-276)
SimpleDeposit._externalLock(address,address,uint256) (contracts/mosaic-alpha-contracts/Deposit/SimpleDeposit.sol#357-363) compares to a boolean constant:
    -require(bool,string)(_checkExternalAddress(_account,_locker) == true,Address not approved) (contracts/mosaic-alpha-
contracts/Deposit/SimpleDeposit.sol#359)
SimpleDeposit._externalUnlock(address,address,uint256) (contracts/mosaic-alpha-contracts/Deposit/SimpleDeposit.sol#365-371) compares to a boolean constant:
    -require(bool,string)(_checkExternalAddress(_account,_locker) == true,Address not approved) (contracts/mosaic-alpha-
contracts/Deposit/SimpleDeposit.sol#367)
Register.addWhitelist(address) (contracts/mosaic-alpha-contracts/Register/Register.sol#130-136) compares to a boolean constant:
    -require(bool,string)(blacklistedAddresses[_address] != true,Address is already blacklisted.) (contracts/mosaic-alpha-
contracts/Register/Register.sol#132)
Register.addWhitelist(address) (contracts/mosaic-alpha-contracts/Register/Register.sol#130-136) compares to a boolean constant:
    -require(bool,string)(whitelistedAddresses[_address] != true,Address is already whitelisted.) (contracts/mosaic-alpha-
contracts/Register/Register.sol#131)
Register.addWhitelistBulk(address[]) (contracts/mosaic-alpha-contracts/Register/Register.sol#142-150) compares to a boolean constant:
    -require(bool,string)(blacklistedAddresses[_address[i]] != true,Address is already blacklisted.) (contracts/mosaic-alpha-
contracts/Register/Register.sol#145)
Register.addWhitelistBulk(address[]) (contracts/mosaic-alpha-contracts/Register/Register.sol#142-150) compares to a boolean constant:
    -require(bool,string)(whitelistedAddresses[_address[i]] != true,Address is already whitelisted.) (contracts/mosaic-alpha-
contracts/Register/Register.sol#144)
Register.addBlacklist(address) (contracts/mosaic-alpha-contracts/Register/Register.sol#156-162) compares to a boolean constant:
    -require(bool,string)(blacklistedAddresses[_address] != true,Address is already blacklisted.) (contracts/mosaic-alpha-
contracts/Register/Register.sol#158)
Register.addBlacklist(address) (contracts/mosaic-alpha-contracts/Register/Register.sol#156-162) compares to a boolean constant:
    -require(bool,string)(whitelistedAddresses[_address] != true,Address is already whitelisted.) (contracts/mosaic-alpha-
contracts/Register/Register.sol#157)
Register.removeWhitelist(address) (contracts/mosaic-alpha-contracts/Register/Register.sol#168-172) compares to a boolean constant:
    -require(bool,string)(whitelistedAddresses[_address] == true,Not in a whitelist!) (contracts/mosaic-alpha-contracts/Register/Register.sol#169)
Affiliate_Storage.pause_state() (contracts/mosaic-alpha-contracts/User/Affiliate_old.sol#33-36) compares to a boolean constant:
    -require(bool,string)(paused == false,Paused.) (contracts/mosaic-alpha-contracts/User/Affiliate_old.sol#34)
SystemWhitelist.addWhitelist(address) (contracts/mosaic-alpha-contracts/helpers/SystemWhitelist.sol#69-75) compares to a boolean constant:
    -require(bool,string)(blacklistedAddresses[_address] != true,Address is already blacklisted.) (contracts/mosaic-alpha-
contracts/helpers/SystemWhitelist.sol#71)
SystemWhitelist.addWhitelist(address) (contracts/mosaic-alpha-contracts/helpers/SystemWhitelist.sol#69-75) compares to a boolean constant:
    -require(bool,string)(whitelistedAddresses[_address] != true,Address is already whitelisted.) (contracts/mosaic-alpha-
contracts/helpers/SystemWhitelist.sol#70)
SystemWhitelist.addWhitelistBulk(address[]) (contracts/mosaic-alpha-contracts/helpers/SystemWhitelist.sol#81-89) compares to a boolean constant:
    -require(bool,string)(blacklistedAddresses[_address[i]] != true,Address is already blacklisted.) (contracts/mosaic-alpha-
contracts/helpers/SystemWhitelist.sol#84)
SystemWhitelist.addWhitelistBulk(address[]) (contracts/mosaic-alpha-contracts/helpers/SystemWhitelist.sol#81-89) compares to a boolean constant:
    -require(bool,string)(whitelistedAddresses[_address[i]] != true,Address is already whitelisted.) (contracts/mosaic-alpha-
contracts/helpers/SystemWhitelist.sol#83)
SystemWhitelist.addBlacklist(address) (contracts/mosaic-alpha-contracts/helpers/SystemWhitelist.sol#95-101) compares to a boolean constant:
    -require(bool,string)(blacklistedAddresses[_address] != true,Address is already blacklisted.) (contracts/mosaic-alpha-
contracts/helpers/SystemWhitelist.sol#97)
SystemWhitelist.addBlacklist(address) (contracts/mosaic-alpha-contracts/helpers/SystemWhitelist.sol#95-101) compares to a boolean constant:
    -require(bool,string)(whitelistedAddresses[_address] != true,Address is already whitelisted.) (contracts/mosaic-alpha-
contracts/helpers/SystemWhitelist.sol#96)
SystemWhitelist.removeWhitelist(address) (contracts/mosaic-alpha-contracts/helpers/SystemWhitelist.sol#107-111) compares to a boolean constant:
    -require(bool,string)(whitelistedAddresses[_address] == true,Not in a whitelist!) (contracts/mosaic-alpha-contracts/helpers/SystemWhitelist.sol#108)
GovernedContractsMM.Live() (contracts/test/ConfigGovernedExample.sol#28-31) compares to a boolean constant:
    -require(bool,string)(Running == true,Emergency stopped.) (contracts/test/ConfigGovernedExample.sol#29)
GovernedContractsMM.onlyManagers() (contracts/test/ConfigGovernedExample.sol#34-40) compares to a boolean constant:
    -ok == true (contracts/test/ConfigGovernedExample.sol#39)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#boolean-equality
```

different-pragma-directives-are-used

```
Different versions of Solidity are used:
    - Version used: ['=0.5.16', '>=0.5.0']
    - =0.5.16 (contracts/mosaic-alpha-contracts/helpers/pancake/PancakeERC20.sol#2)
    - =0.5.16 (contracts/mosaic-alpha-contracts/helpers/pancake/PancakeFactory.sol#2)
    - =0.5.16 (contracts/mosaic-alpha-contracts/helpers/pancake/PancakePair.sol#2)
    - >=0.5.0 (contracts/mosaic-alpha-contracts/helpers/pancake/interfaces/IERC20.sol#3)
    - >=0.5.0 (contracts/mosaic-alpha-contracts/helpers/pancake/interfaces/IPancakeCallee.sol#3)
    - >=0.5.0 (contracts/mosaic-alpha-contracts/helpers/pancake/interfaces/IPancakeERC20.sol#3)
    - >=0.5.0 (contracts/mosaic-alpha-contracts/helpers/pancake/interfaces/IPancakeFactory.sol#3)
    - >=0.5.0 (contracts/mosaic-alpha-contracts/helpers/pancake/interfaces/IPancakePair.sol#3)
    - >=0.5.0 (contracts/mosaic-alpha-contracts/helpers/pancake/libraries/Math.sol#2)
    - >=0.5.0 (contracts/mosaic-alpha-contracts/helpers/pancake/libraries/SafeMath.sol#2)
    - >=0.5.0 (contracts/mosaic-alpha-contracts/helpers/pancake/libraries/UQ112x112.sol#2)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used

Different versions of Solidity are used:
    - Version used: ['=0.6.12', '>=0.5.0', '>=0.5.0<0.7.0', '>=0.6.0', '>=0.6.2']
    - =0.6.12 (contracts/mosaic-alpha-contracts/helpers/pancake/PancakeRouter.sol#2)
    - >=0.5.0 (contracts/mosaic-alpha-contracts/helpers/pancake/interfaces/IERC20.sol#3)
    - >=0.5.0 (contracts/mosaic-alpha-contracts/helpers/pancake/interfaces/IPancakeFactory.sol#3)
    - >=0.5.0 (contracts/mosaic-alpha-contracts/helpers/pancake/interfaces/IPancakePair.sol#3)
    - >=0.6.2 (contracts/mosaic-alpha-contracts/helpers/pancake/interfaces/IPancakeRouter01.sol#4)
    - >=0.6.2 (contracts/mosaic-alpha-contracts/helpers/pancake/interfaces/IPancakeRouter02.sol#3)
    - >=0.5.0 (contracts/mosaic-alpha-contracts/helpers/pancake/interfaces/IWETH.sol#2)
    - >=0.5.0<0.7.0 (contracts/mosaic-alpha-contracts/helpers/pancake/libraries/PancakeLibrary.sol#2)
    - >=0.5.0 (contracts/mosaic-alpha-contracts/helpers/pancake/libraries/SafeMath.sol#2)
    - >=0.6.0 (contracts/mosaic-alpha-contracts/helpers/pancake/libraries/TransferHelper.sol#2)
```

```
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used

Different versions of Solidity are used:
    - Version used: ['>=0.4.22<0.9.0', '>=0.4.24<0.9', '^0.8.0', '^0.8.17']
    - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/IERC20.sol#4)
    - ^0.8.0 (node_modules/@openzeppelin/contracts/utils/math/SafeCast.sol#4)
    - ^0.8.17 (contracts/mosaic-alpha-contracts/Interfaces/IFees.sol#2)
    - ^0.8.17 (contracts/mosaic-alpha-contracts/Interfaces/IGoverned.sol#2)
    - >=0.4.24<0.9 (contracts/mosaic-alpha-contracts/Interfaces/IMosaicOracle.sol#2)
    - ^0.8.17 (contracts/mosaic-alpha-contracts/Interfaces/IPool.sol#2)
    - ^0.8.17 (contracts/mosaic-alpha-contracts/Interfaces/IVault.sol#2)
    - ^0.8.17 (contracts/mosaic-alpha-contracts/Pool/Math.sol#2)
    - ^0.8.17 (contracts/mosaic-alpha-contracts/Pool/Pool.sol#2)
    - >=0.4.22<0.9.0 (node_modules/hardhat/console.sol#2)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used

Different versions of Solidity are used:
    - Version used: ['>=0.4.24<0.9', '^0.7.0||^0.8.0', '^0.8.0', '^0.8.1', '^0.8.17']
    - ^0.8.0 (node_modules/@openzeppelin/contracts/security/ReentrancyGuard.sol#4)
    - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/IERC20.sol#4)
    - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/extensions/draft-IERC20Permit.sol#4)
    - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#4)
    - ^0.8.1 (node_modules/@openzeppelin/contracts/utils/Address.sol#4)
    - ^0.8.0 (node_modules/@openzeppelin/contracts/utils/math/SafeCast.sol#4)
    - ^0.8.17 (contracts/mosaic-alpha-contracts/Governance/Governance.sol#2)
    - ^0.8.17 (contracts/mosaic-alpha-contracts/Governance/Governed.sol#2)
    - ^0.8.17 (contracts/mosaic-alpha-contracts/Interfaces/IAffiliate.sol#2)
    - ^0.7.0||^0.8.0 (contracts/mosaic-alpha-contracts/Interfaces/IERC3156FlashBorrower.sol#2)
    - ^0.7.0||^0.8.0 (contracts/mosaic-alpha-contracts/Interfaces/IERC3156FlashLender.sol#2)
    - ^0.8.17 (contracts/mosaic-alpha-contracts/Interfaces/IFeeBurner.sol#2)
    - ^0.8.17 (contracts/mosaic-alpha-contracts/Interfaces/IFees.sol#2)
    - ^0.8.17 (contracts/mosaic-alpha-contracts/Interfaces/IGovernance.sol#2)
    - ^0.8.17 (contracts/mosaic-alpha-contracts/Interfaces/IGoverned.sol#2)
    - >=0.4.24<0.9 (contracts/mosaic-alpha-contracts/Interfaces/IMosaicOracle.sol#2)
    - ^0.8.17 (contracts/mosaic-alpha-contracts/Interfaces/IPool.sol#2)
    - ^0.8.17 (contracts/mosaic-alpha-contracts/Interfaces/IRegister.sol#2)
    - ^0.8.17 (contracts/mosaic-alpha-contracts/Interfaces/ISwaps.sol#2)
    - ^0.8.17 (contracts/mosaic-alpha-contracts/Interfaces/IUserProfile.sol#2)
    - ^0.8.17 (contracts/mosaic-alpha-contracts/Interfaces/IVault.sol#2)
    - ^0.8.17 (contracts/mosaic-alpha-contracts/Vault/Vault.sol#2)
    - ^0.8.17 (contracts/mosaic-alpha-contracts/helpers/IERC20Burnable.sol#2)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used

Different versions of Solidity are used:
    - Version used: ['^0.8.0', '^0.8.1', '^0.8.17']
    - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/IERC20.sol#4)
    - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/extensions/draft-IERC20Permit.sol#4)
    - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#4)
    - ^0.8.1 (node_modules/@openzeppelin/contracts/utils/Address.sol#4)
    - ^0.8.0 (node_modules/@openzeppelin/contracts/utils/math/SafeCast.sol#4)
    - ^0.8.17 (contracts/mosaic-alpha-contracts/Fees/Fees.sol#2)
    - ^0.8.17 (contracts/mosaic-alpha-contracts/Governance/Governance.sol#2)
    - ^0.8.17 (contracts/mosaic-alpha-contracts/Governance/Governed.sol#2)
    - ^0.8.17 (contracts/mosaic-alpha-contracts/Interfaces/IAffiliate.sol#2)
    - ^0.8.17 (contracts/mosaic-alpha-contracts/Interfaces/IFees.sol#2)
    - ^0.8.17 (contracts/mosaic-alpha-contracts/Interfaces/IFeesClaimLimitAdapter.sol#2)
    - ^0.8.17 (contracts/mosaic-alpha-contracts/Interfaces/IGovernance.sol#2)
    - ^0.8.17 (contracts/mosaic-alpha-contracts/Interfaces/IGoverned.sol#2)
    - ^0.8.17 (contracts/mosaic-alpha-contracts/Interfaces/IPool.sol#2)
    - ^0.8.17 (contracts/mosaic-alpha-contracts/Interfaces/IUserProfile.sol#2)
    - ^0.8.17 (contracts/mosaic-alpha-contracts/Interfaces/IVault.sol#2)
    - ^0.8.0 (contracts/mosaic-alpha-contracts/helpers/Arrays.sol#4)
    - ^0.8.17 (contracts/mosaic-alpha-contracts/helpers/Controllable.sol#3)
    - ^0.8.0 (contracts/mosaic-alpha-contracts/helpers/Ownable.sol#4)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used

Different versions of Solidity are used:
    - Version used: ['>=0.4.22<0.9.0', '>=0.4.24<0.9', '>=0.4.24<0.9.0', '>=0.5.0', '>=0.5.0<0.9.0', '>=0.5.0<7.0.0', '>=0.6.0', '>=0.6.2',
'^0.7.0||^0.8.0', '^0.8.0', '^0.8.1', '^0.8.17', '^0.8.9']
    - ^0.8.0 (node_modules/@openzeppelin/contracts/access/AccessControl.sol#4)
    - ^0.8.0 (node_modules/@openzeppelin/contracts/access/IAccessControl.sol#4)
    - ^0.8.0 (node_modules/@openzeppelin/contracts/access/Ownable.sol#4)
    - ^0.8.0 (node_modules/@openzeppelin/contracts/security/Pausable.sol#4)
    - ^0.8.0 (node_modules/@openzeppelin/contracts/security/ReentrancyGuard.sol#4)
    - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#4)
    - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/IERC20.sol#4)
    - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/extensions/ERC20Burnable.sol#4)
    - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/extensions/IERC20Metadata.sol#4)
    - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/extensions/draft-IERC20Permit.sol#4)
    - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#4)
    - ^0.8.1 (node_modules/@openzeppelin/contracts/utils/Address.sol#4)
    - ^0.8.0 (node_modules/@openzeppelin/contracts/utils/Context.sol#4)
    - ^0.8.0 (node_modules/@openzeppelin/contracts/utils/Counters.sol#4)
    - ^0.8.0 (node_modules/@openzeppelin/contracts/utils/Strings.sol#4)
    - ^0.8.0 (node_modules/@openzeppelin/contracts/utils/introspection/ERC165.sol#4)
    - ^0.8.0 (node_modules/@openzeppelin/contracts/utils/introspection/IERC165.sol#4)
    - ^0.8.0 (node_modules/@openzeppelin/contracts/utils/math/SafeCast.sol#4)
    - ^0.8.0 (contracts/.deps/npm/@openzeppelin/contracts/token/ERC20/IERC20.sol#4)
    - ^0.8.1 (contracts/.deps/npm/@openzeppelin/contracts/utils/Address.sol#4)
    - ^0.8.9 (contracts/mosaic-alpha-bot/Interfaces/Interfaces/IERC20.sol#2)
    - ^0.8.9 (contracts/mosaic-alpha-bot/Interfaces/Interfaces/IPancakeRouter.sol#2)
    - ^0.8.9 (contracts/mosaic-alpha-bot/Interfaces/Interfaces/IPool.sol#2)
    - ^0.8.17 (contracts/mosaic-alpha-bot/Interfaces/Interfaces/IQueryApi.sol#2)
    - ^0.8.9 (contracts/mosaic-alpha-bot/Interfaces/Interfaces/ITradingBot.sol#2)
    - ^0.8.17 (contracts/mosaic-alpha-bot/Interfaces/Interfaces/IVault.sol#2)
    - ^0.8.17 (contracts/mosaic-alpha-bot/Interfaces/TradingBot.sol#2)
    - ^0.8.17 (contracts/mosaic-alpha-contracts/Deposit/SimpleDeposit.sol#2)
    - ^0.8.17 (contracts/mosaic-alpha-contracts/Fees/FeeBurner.sol#2)
    - ^0.8.17 (contracts/mosaic-alpha-contracts/Governance/Governance.sol#2)
    - ^0.8.17 (contracts/mosaic-alpha-contracts/Governance/Governed.sol#2)
    - ^0.8.17 (contracts/mosaic-alpha-contracts/Interfaces/IAffiliate.sol#2)
    - ^0.8.17 (contracts/mosaic-alpha-contracts/Interfaces/IAffiliateBooster.sol#2)
    - ^0.8.17 (contracts/mosaic-alpha-contracts/Interfaces/IAffiliateBoosterController.sol#2)
    - ^0.7.0||^0.8.0 (contracts/mosaic-alpha-contracts/Interfaces/IERC3156FlashBorrower.sol#2)
    - ^0.7.0||^0.8.0 (contracts/mosaic-alpha-contracts/Interfaces/IERC3156FlashLender.sol#2)
    - ^0.8.17 (contracts/mosaic-alpha-contracts/Interfaces/IFeeBurner.sol#2)
    - ^0.8.17 (contracts/mosaic-alpha-contracts/Interfaces/IFees.sol#2)
    - ^0.8.17 (contracts/mosaic-alpha-contracts/Interfaces/IFeesClaimLimitAdapter.sol#2)
    - ^0.8.17 (contracts/mosaic-alpha-contracts/Interfaces/IGovernance.sol#2)
    - ^0.8.17 (contracts/mosaic-alpha-contracts/Interfaces/IGoverned.sol#2)
    - >=0.4.24<0.9 (contracts/mosaic-alpha-contracts/Interfaces/IMosaicOracle.sol#2)
    - ^0.8.17 (contracts/mosaic-alpha-contracts/Interfaces/IPool.sol#2)
    - ^0.8.17 (contracts/mosaic-alpha-contracts/Interfaces/IQueryApi.sol#2)
    - ^0.8.17 (contracts/mosaic-alpha-contracts/Interfaces/IRegister.sol#2)
    - ^0.8.17 (contracts/mosaic-alpha-contracts/Interfaces/ISimpleDeposit.sol#2)
    - ^0.8.17 (contracts/mosaic-alpha-contracts/Interfaces/ISwaps.sol#2)
    - ^0.8.17 (contracts/mosaic-alpha-contracts/Interfaces/IUserProfile.sol#2)
```

```
    - ^0.8.17 (contracts/mosaic-alpha-contracts/Interfaces/IVault.sol#2)
    - ^0.8.17 (contracts/mosaic-alpha-contracts/Interfaces/IVaultDelegated.sol#2)
    - ^0.8.17 (contracts/mosaic-alpha-contracts/Libraries/PoolLibrary.sol#2)
    - >=0.4.24<0.9.0 (contracts/mosaic-alpha-contracts/Oracle/Initializable.sol#3)
    - ^0.8.17 (contracts/mosaic-alpha-contracts/Oracle/SimpleMosaicOracle.sol#2)
    - ^0.8.17 (contracts/mosaic-alpha-contracts/Pool/Math.sol#2)
    - ^0.8.17 (contracts/mosaic-alpha-contracts/Pool/Pool.sol#2)
    - ^0.8.17 (contracts/mosaic-alpha-contracts/Pool/PoolFactory.sol#2)
    - ^0.8.17 (contracts/mosaic-alpha-contracts/Query/QueryApi.sol#2)
    - ^0.8.17 (contracts/mosaic-alpha-contracts/Register/Register.sol#2)
    - ^0.8.17 (contracts/mosaic-alpha-contracts/User/Affiliate.sol#2)
    - ^0.8.17 (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#2)
    - ^0.8.17 (contracts/mosaic-alpha-contracts/User/Affiliate_old.sol#2)
    - ^0.8.17 (contracts/mosaic-alpha-contracts/User/SimpleAffiliateBoosterController.sol#2)
    - ^0.8.17 (contracts/mosaic-alpha-contracts/User/UserProfile.sol#2)
    - ^0.8.17 (contracts/mosaic-alpha-contracts/Vault/FlashLoans.sol#2)
    - ^0.8.17 (contracts/mosaic-alpha-contracts/Vault/Swaps.sol#2)
    - ^0.8.0 (contracts/mosaic-alpha-contracts/helpers/Arrays.sol#4)
    - ^0.8.17 (contracts/mosaic-alpha-contracts/helpers/Controllable.sol#3)
    - ^0.8.17 (contracts/mosaic-alpha-contracts/helpers/IERC20Burnable.sol#2)
    - ^0.8.17 (contracts/mosaic-alpha-contracts/helpers/IEntropy.sol#2)
    - ^0.8.17 (contracts/mosaic-alpha-contracts/helpers/ISystemRole.sol#2)
    - ^0.8.17 (contracts/mosaic-alpha-contracts/helpers/ISystemWhitelist.sol#2)
    - ^0.8.0 (contracts/mosaic-alpha-contracts/helpers/Ownable.sol#4)
    - ^0.8.17 (contracts/mosaic-alpha-contracts/helpers/SimpleEntropy.sol#3)
    - ^0.8.17 (contracts/mosaic-alpha-contracts/helpers/SystemRole.sol#2)
    - ^0.8.17 (contracts/mosaic-alpha-contracts/helpers/SystemWhitelist.sol#2)
    - >=0.5.0 (contracts/mosaic-alpha-contracts/helpers/pancake/interfaces/IERC20.sol#3)
    - >=0.5.0 (contracts/mosaic-alpha-contracts/helpers/pancake/interfaces/IPancakeCallee.sol#3)
    - >=0.5.0 (contracts/mosaic-alpha-contracts/helpers/pancake/interfaces/IPancakeERC20.sol#3)
    - >=0.5.0 (contracts/mosaic-alpha-contracts/helpers/pancake/interfaces/IPancakeFactory.sol#3)
    - >=0.5.0<0.9.0 (contracts/mosaic-alpha-contracts/helpers/pancake/interfaces/IPancakeOracle.sol#2)
    - >=0.5.0 (contracts/mosaic-alpha-contracts/helpers/pancake/interfaces/IPancakePair.sol#3)
    - >=0.6.2 (contracts/mosaic-alpha-contracts/helpers/pancake/interfaces/IPancakeRouter01.sol#4)
    - >=0.6.2 (contracts/mosaic-alpha-contracts/helpers/pancake/interfaces/IPancakeRouter02.sol#3)
    - >=0.5.0 (contracts/mosaic-alpha-contracts/helpers/pancake/interfaces/IWETH.sol#2)
    - >=0.5.0 (contracts/mosaic-alpha-contracts/helpers/pancake/libraries/Babylonian.sol#3)
    - >=0.5.0 (contracts/mosaic-alpha-contracts/helpers/pancake/libraries/BitMath.sol#2)
    - >=0.5.0 (contracts/mosaic-alpha-contracts/helpers/pancake/libraries/FixedPoint.sol#2)
    - >=0.5.0 (contracts/mosaic-alpha-contracts/helpers/pancake/libraries/FullMath.sol#2)
    - >=0.5.0 (contracts/mosaic-alpha-contracts/helpers/pancake/libraries/Math.sol#2)
    - >=0.5.0<7.0.0 (contracts/mosaic-alpha-contracts/helpers/pancake/libraries/PancakeOracleLibrary.sol#2)
    - >=0.5.0 (contracts/mosaic-alpha-contracts/helpers/pancake/libraries/SafeMath.sol#2)
    - >=0.6.0 (contracts/mosaic-alpha-contracts/helpers/pancake/libraries/TransferHelper.sol#2)
    - >=0.5.0 (contracts/mosaic-alpha-contracts/helpers/pancake/libraries/UQ112x112.sol#2)
    - ^0.8.17 (contracts/test/ConfigGovernedExample.sol#2)
    - ^0.8.17 (contracts/test/TestToken.sol#3)
    - >=0.4.22<0.9.0 (node_modules/hardhat/console.sol#2)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used
```

costly-operations-inside-a-loop

```
ReentrancyGuard.nonReentrant() (node_modules/@openzeppelin/contracts/security/ReentrancyGuard.sol#50-62) has costly operations inside a loop:
    - _status = _ENTERED (node_modules/@openzeppelin/contracts/security/ReentrancyGuard.sol#55)
ReentrancyGuard.nonReentrant() (node_modules/@openzeppelin/contracts/security/ReentrancyGuard.sol#50-62) has costly operations inside a loop:
    - _status = _NOT_ENTERED (node_modules/@openzeppelin/contracts/security/ReentrancyGuard.sol#61)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#costly-operations-inside-a-loop
```

dead-code

```
TransferHelper.safeApprove(address,address,uint256) (contracts/mosaic-alpha-contracts/helpers/pancake/libraries/TransferHelper.sol#6-10) is never used and
should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

FixedPoint.ln(int256) (contracts/mosaic-alpha-contracts/Pool/Math.sol#408-416) is never used and should be removed
FixedPoint.log(int256,int256) (contracts/mosaic-alpha-contracts/Pool/Math.sol#381-403) is never used and should be removed
FixedPoint.powDown(uint256,uint256) (contracts/mosaic-alpha-contracts/Pool/Math.sol#69-89) is never used and should be removed
Pool._checkOwner() (contracts/mosaic-alpha-contracts/Pool/Pool.sol#121-123) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Governed.notLiveFun() (contracts/mosaic-alpha-contracts/Governance/Governed.sol#50-52) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Controllable._isController(address) (contracts/mosaic-alpha-contracts/helpers/Controllable.sol#53-55) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Address._revert(bytes,string) (contracts/.deps/npm/@openzeppelin/contracts/utils/Address.sol#231-243) is never used and should be removed
Address.functionCall(address,bytes) (contracts/.deps/npm/@openzeppelin/contracts/utils/Address.sol#85-87) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (contracts/.deps/npm/@openzeppelin/contracts/utils/Address.sol#114-120) is never used and should be
removed
Address.functionDelegateCall(address,bytes) (contracts/.deps/npm/@openzeppelin/contracts/utils/Address.sol#170-172) is never used and should be removed
Address.functionDelegateCall(address,bytes,string) (contracts/.deps/npm/@openzeppelin/contracts/utils/Address.sol#180-187) is never used and should be
removed
Address.functionStaticCall(address,bytes) (contracts/.deps/npm/@openzeppelin/contracts/utils/Address.sol#145-147) is never used and should be removed
Address.functionStaticCall(address,bytes,string) (contracts/.deps/npm/@openzeppelin/contracts/utils/Address.sol#155-162) is never used and should be
removed
Address.sendValue(address,uint256) (contracts/.deps/npm/@openzeppelin/contracts/utils/Address.sol#60-65) is never used and should be removed
Address.verifyCallResultFromTarget(address,bool,bytes,string) (contracts/.deps/npm/@openzeppelin/contracts/utils/Address.sol#195-211) is never used and
should be removed
AffiliateBooster._setFeeTokenBurnable(bool) (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#297-299) is never used and should be removed
Arrays.findUpperBound(uint256[],uint256) (contracts/mosaic-alpha-contracts/helpers/Arrays.sol#19-48) is never used and should be removed
Babylonian.sqrt(uint256) (contracts/mosaic-alpha-contracts/helpers/pancake/libraries/Babylonian.sol#10-52) is never used and should be removed
BitMath.leastSignificantBit(uint256) (contracts/mosaic-alpha-contracts/helpers/pancake/libraries/BitMath.sol#44-89) is never used and should be removed
BitMath.mostSignificantBit(uint256) (contracts/mosaic-alpha-contracts/helpers/pancake/libraries/BitMath.sol#7-39) is never used and should be removed
FixedPoint.decode(FixedPoint.uq112x112) (contracts/mosaic-alpha-contracts/helpers/pancake/libraries/FixedPoint.sol#38-40) is never used and should be
removed
FixedPoint.decode144(FixedPoint.uq144x112) (contracts/mosaic-alpha-contracts/helpers/pancake/libraries/FixedPoint.sol#43-45) is never used and should be
removed
FixedPoint.divuq(FixedPoint.uq112x112,FixedPoint.uq112x112) (contracts/mosaic-alpha-contracts/helpers/pancake/libraries/FixedPoint.sol#93-107) is never
used and should be removed
FixedPoint.encode(uint112) (contracts/mosaic-alpha-contracts/helpers/pancake/libraries/FixedPoint.sol#28-30) is never used and should be removed
FixedPoint.encode144(uint144) (contracts/mosaic-alpha-contracts/helpers/pancake/libraries/FixedPoint.sol#33-35) is never used and should be removed
FixedPoint.fraction(uint256,uint256) (contracts/mosaic-alpha-contracts/helpers/pancake/libraries/FixedPoint.sol#111-124) is never used and should be
removed
FixedPoint.mul(FixedPoint.uq112x112,uint256) (contracts/mosaic-alpha-contracts/helpers/pancake/libraries/FixedPoint.sol#49-53) is never used and should be
removed
FixedPoint.muli(FixedPoint.uq112x112,int256) (contracts/mosaic-alpha-contracts/helpers/pancake/libraries/FixedPoint.sol#57-61) is never used and should be
removed
FixedPoint.muluq(FixedPoint.uq112x112,FixedPoint.uq112x112) (contracts/mosaic-alpha-contracts/helpers/pancake/libraries/FixedPoint.sol#65-90) is never used
and should be removed
FixedPoint.reciprocal(FixedPoint.uq112x112) (contracts/mosaic-alpha-contracts/helpers/pancake/libraries/FixedPoint.sol#129-133) is never used and should be
removed
FixedPoint.sqrt(FixedPoint.uq112x112) (contracts/mosaic-alpha-contracts/helpers/pancake/libraries/FixedPoint.sol#137-145) is never used and should be
removed
```

```
FullMath.fullDiv(uint256,uint256,uint256) (contracts/mosaic-alpha-contracts/helpers/pancake/libraries/FullMath.sol#14-35) is never used and should be
removed
FullMath.fullMul(uint256,uint256) (contracts/mosaic-alpha-contracts/helpers/pancake/libraries/FullMath.sol#7-12) is never used and should be removed
FullMath.mulDiv(uint256,uint256,uint256) (contracts/mosaic-alpha-contracts/helpers/pancake/libraries/FullMath.sol#37-52) is never used and should be
removed
Math.min(uint256,uint256) (contracts/mosaic-alpha-contracts/helpers/pancake/libraries/Math.sol#6-8) is never used and should be removed
Math.sqrt(uint256) (contracts/mosaic-alpha-contracts/helpers/pancake/libraries/Math.sol#11-22) is never used and should be removed
PancakeOracleLibrary.currentBlockTimestamp() (contracts/mosaic-alpha-contracts/helpers/pancake/libraries/PancakeOracleLibrary.sol#11-13) is never used and
should be removed
PancakeOracleLibrary.currentCumulativePrices(address) (contracts/mosaic-alpha-contracts/helpers/pancake/libraries/PancakeOracleLibrary.sol#16-34) is never
used and should be removed
PoolLibrary._checkWeightsValidity(uint32[]) (contracts/mosaic-alpha-contracts/Libraries/PoolLibrary.sol#9-15) is never used and should be removed
SafeMath.add(uint256,uint256) (contracts/mosaic-alpha-contracts/helpers/pancake/libraries/SafeMath.sol#7-9) is never used and should be removed
SafeMath.mul(uint256,uint256) (contracts/mosaic-alpha-contracts/helpers/pancake/libraries/SafeMath.sol#15-17) is never used and should be removed
SafeMath.sub(uint256,uint256) (contracts/mosaic-alpha-contracts/helpers/pancake/libraries/SafeMath.sol#11-13) is never used and should be removed
TransferHelper.safeTransfer(address,address,uint256) (contracts/mosaic-alpha-contracts/helpers/pancake/libraries/TransferHelper.sol#12-16) is never used
and should be removed
TransferHelper.safeTransferETH(address,uint256) (contracts/mosaic-alpha-contracts/helpers/pancake/libraries/TransferHelper.sol#24-27) is never used and
should be removed
TransferHelper.safeTransferFrom(address,address,address,uint256) (contracts/mosaic-alpha-contracts/helpers/pancake/libraries/TransferHelper.sol#18-22) is
never used and should be removed
UQ112x112.encode(uint112) (contracts/mosaic-alpha-contracts/helpers/pancake/libraries/UQ112x112.sol#11-13) is never used and should be removed
UQ112x112.uqdiv(uint224,uint112) (contracts/mosaic-alpha-contracts/helpers/pancake/libraries/UQ112x112.sol#16-18) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
```

incorrect-versions-of-solidity

```
Pragma version>=0.5.0 (contracts/mosaic-alpha-contracts/helpers/pancake/interfaces/IERC20.sol#3) allows old versions
Pragma version>=0.5.0 (contracts/mosaic-alpha-contracts/helpers/pancake/interfaces/IPancakeCallee.sol#3) allows old versions
Pragma version>=0.5.0 (contracts/mosaic-alpha-contracts/helpers/pancake/interfaces/IPancakeERC20.sol#3) allows old versions
Pragma version>=0.5.0 (contracts/mosaic-alpha-contracts/helpers/pancake/interfaces/IPancakeFactory.sol#3) allows old versions
Pragma version>=0.5.0 (contracts/mosaic-alpha-contracts/helpers/pancake/interfaces/IPancakePair.sol#3) allows old versions
Pragma version>=0.5.0 (contracts/mosaic-alpha-contracts/helpers/pancake/libraries/Math.sol#2) allows old versions
Pragma version>=0.5.0 (contracts/mosaic-alpha-contracts/helpers/pancake/libraries/SafeMath.sol#2) allows old versions
Pragma version>=0.5.0 (contracts/mosaic-alpha-contracts/helpers/pancake/libraries/UQ112x112.sol#2) allows old versions
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Pragma version>=0.6.2 (contracts/mosaic-alpha-contracts/helpers/pancake/interfaces/IPancakeRouter01.sol#4) allows old versions
Pragma version>=0.6.2 (contracts/mosaic-alpha-contracts/helpers/pancake/interfaces/IPancakeRouter02.sol#3) allows old versions
Pragma version>=0.5.0 (contracts/mosaic-alpha-contracts/helpers/pancake/interfaces/IWETH.sol#2) allows old versions
Pragma version>=0.5.0<0.7.0 (contracts/mosaic-alpha-contracts/helpers/pancake/libraries/PancakeLibrary.sol#2) allows old versions
Pragma version>=0.6.0 (contracts/mosaic-alpha-contracts/helpers/pancake/libraries/TransferHelper.sol#2) allows old versions
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/IERC20.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/utils/math/SafeCast.sol#4) allows old versions
Pragma version^0.8.17 (contracts/mosaic-alpha-contracts/Interfaces/IFees.sol#2) necessitates a version too recent to be trusted. Consider deploying with
0.6.12/0.7.6/0.8.7
Pragma version^0.8.17 (contracts/mosaic-alpha-contracts/Interfaces/IGoverned.sol#2) necessitates a version too recent to be trusted. Consider deploying
with 0.6.12/0.7.6/0.8.7
Pragma version>=0.4.24<0.9 (contracts/mosaic-alpha-contracts/Interfaces/IMosaicOracle.sol#2) allows old versions
Pragma version^0.8.17 (contracts/mosaic-alpha-contracts/Interfaces/IPool.sol#2) necessitates a version too recent to be trusted. Consider deploying with
0.6.12/0.7.6/0.8.7
Pragma version^0.8.17 (contracts/mosaic-alpha-contracts/Interfaces/IVault.sol#2) necessitates a version too recent to be trusted. Consider deploying with
0.6.12/0.7.6/0.8.7
Pragma version^0.8.17 (contracts/mosaic-alpha-contracts/Pool/Math.sol#2) necessitates a version too recent to be trusted. Consider deploying with
0.6.12/0.7.6/0.8.7
Pragma version^0.8.17 (contracts/mosaic-alpha-contracts/Pool/Pool.sol#2) necessitates a version too recent to be trusted. Consider deploying with
0.6.12/0.7.6/0.8.7
Pragma version>=0.4.22<0.9.0 (node_modules/hardhat/console.sol#2) is too complex
solc-0.8.19 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/security/ReentrancyGuard.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/extensions/draft-IERC20Permit.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#4) allows old versions
Pragma version^0.8.1 (node_modules/@openzeppelin/contracts/utils/Address.sol#4) allows old versions
Pragma version^0.8.17 (contracts/mosaic-alpha-contracts/Governance/Governance.sol#2) necessitates a version too recent to be trusted. Consider deploying
with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.17 (contracts/mosaic-alpha-contracts/Governance/Governed.sol#2) necessitates a version too recent to be trusted. Consider deploying with
0.6.12/0.7.6/0.8.7
Pragma version^0.8.17 (contracts/mosaic-alpha-contracts/Interfaces/IAffiliate.sol#2) necessitates a version too recent to be trusted. Consider deploying
with 0.6.12/0.7.6/0.8.7
Pragma version^0.7.0||^0.8.0 (contracts/mosaic-alpha-contracts/Interfaces/IERC3156FlashBorrower.sol#2) is too complex
Pragma version^0.7.0||^0.8.0 (contracts/mosaic-alpha-contracts/Interfaces/IERC3156FlashLender.sol#2) is too complex
Pragma version^0.8.17 (contracts/mosaic-alpha-contracts/Interfaces/IFeeBurner.sol#2) necessitates a version too recent to be trusted. Consider deploying
with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.17 (contracts/mosaic-alpha-contracts/Interfaces/IGovernance.sol#2) necessitates a version too recent to be trusted. Consider deploying
with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.17 (contracts/mosaic-alpha-contracts/Interfaces/IRegister.sol#2) necessitates a version too recent to be trusted. Consider deploying
with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.17 (contracts/mosaic-alpha-contracts/Interfaces/ISwaps.sol#2) necessitates a version too recent to be trusted. Consider deploying with
0.6.12/0.7.6/0.8.7
Pragma version^0.8.17 (contracts/mosaic-alpha-contracts/Interfaces/IUserProfile.sol#2) necessitates a version too recent to be trusted. Consider deploying
with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.17 (contracts/mosaic-alpha-contracts/Vault/Vault.sol#2) necessitates a version too recent to be trusted. Consider deploying with
0.6.12/0.7.6/0.8.7
Pragma version^0.8.17 (contracts/mosaic-alpha-contracts/helpers/IERC20Burnable.sol#2) necessitates a version too recent to be trusted. Consider deploying
with 0.6.12/0.7.6/0.8.7
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Pragma version^0.8.17 (contracts/mosaic-alpha-contracts/Fees/Fees.sol#2) necessitates a version too recent to be trusted. Consider deploying with
0.6.12/0.7.6/0.8.7
Pragma version^0.8.17 (contracts/mosaic-alpha-contracts/Interfaces/IFeesClaimLimitAdapter.sol#2) necessitates a version too recent to be trusted. Consider
deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.0 (contracts/mosaic-alpha-contracts/helpers/Arrays.sol#4) allows old versions
Pragma version^0.8.17 (contracts/mosaic-alpha-contracts/helpers/Controllable.sol#3) necessitates a version too recent to be trusted. Consider deploying
with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.0 (contracts/mosaic-alpha-contracts/helpers/Ownable.sol#4) allows old versions
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/access/AccessControl.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/access/IAccessControl.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/access/Ownable.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/security/Pausable.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/extensions/ERC20Burnable.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/extensions/IERC20Metadata.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/utils/Context.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/utils/Counters.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/utils/Strings.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/utils/introspection/ERC165.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/utils/introspection/IERC165.sol#4) allows old versions
Pragma version^0.8.0 (contracts/.deps/npm/@openzeppelin/contracts/token/ERC20/IERC20.sol#4) allows old versions
Pragma version^0.8.1 (contracts/.deps/npm/@openzeppelin/contracts/utils/Address.sol#4) allows old versions
Pragma version^0.8.9 (contracts/mosaic-alpha-bot/Interfaces/Interfaces/IERC20.sol#2) necessitates a version too recent to be trusted. Consider deploying
```

```
with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.9 (contracts/mosaic-alpha-bot/Interfaces/Interfaces/IPancakeRouter.sol#2) necessitates a version too recent to be trusted. Consider
deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.9 (contracts/mosaic-alpha-bot/Interfaces/Interfaces/IPool.sol#2) necessitates a version too recent to be trusted. Consider deploying
with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.17 (contracts/mosaic-alpha-bot/Interfaces/Interfaces/IQueryApi.sol#2) necessitates a version too recent to be trusted. Consider
deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.9 (contracts/mosaic-alpha-bot/Interfaces/Interfaces/ITradingBot.sol#2) necessitates a version too recent to be trusted. Consider
deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.17 (contracts/mosaic-alpha-bot/Interfaces/Interfaces/IVault.sol#2) necessitates a version too recent to be trusted. Consider deploying
with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.17 (contracts/mosaic-alpha-bot/Interfaces/TradingBot.sol#2) necessitates a version too recent to be trusted. Consider deploying with
0.6.12/0.7.6/0.8.7
Pragma version^0.8.17 (contracts/mosaic-alpha-contracts/Deposit/SimpleDeposit.sol#2) necessitates a version too recent to be trusted. Consider deploying
with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.17 (contracts/mosaic-alpha-contracts/Fees/FeeBurner.sol#2) necessitates a version too recent to be trusted. Consider deploying with
0.6.12/0.7.6/0.8.7
Pragma version^0.8.17 (contracts/mosaic-alpha-contracts/Interfaces/IAffiliateBooster.sol#2) necessitates a version too recent to be trusted. Consider
deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.17 (contracts/mosaic-alpha-contracts/Interfaces/IAffiliateBoosterController.sol#2) necessitates a version too recent to be trusted.
Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.17 (contracts/mosaic-alpha-contracts/Interfaces/IQueryApi.sol#2) necessitates a version too recent to be trusted. Consider deploying
with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.17 (contracts/mosaic-alpha-contracts/Interfaces/ISimpleDeposit.sol#2) necessitates a version too recent to be trusted. Consider
deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.17 (contracts/mosaic-alpha-contracts/Interfaces/IVaultDelegated.sol#2) necessitates a version too recent to be trusted. Consider
deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.17 (contracts/mosaic-alpha-contracts/Libraries/PoolLibrary.sol#2) necessitates a version too recent to be trusted. Consider deploying
with 0.6.12/0.7.6/0.8.7
Pragma version>=0.4.24<0.9.0 (contracts/mosaic-alpha-contracts/Oracle/Initializable.sol#3) is too complex
Pragma version^0.8.17 (contracts/mosaic-alpha-contracts/Oracle/SimpleMosaicOracle.sol#2) necessitates a version too recent to be trusted. Consider
deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.17 (contracts/mosaic-alpha-contracts/Pool/PoolFactory.sol#2) necessitates a version too recent to be trusted. Consider deploying with
0.6.12/0.7.6/0.8.7
Pragma version^0.8.17 (contracts/mosaic-alpha-contracts/Query/QueryApi.sol#2) necessitates a version too recent to be trusted. Consider deploying with
0.6.12/0.7.6/0.8.7
Pragma version^0.8.17 (contracts/mosaic-alpha-contracts/Register/Register.sol#2) necessitates a version too recent to be trusted. Consider deploying with
0.6.12/0.7.6/0.8.7
Pragma version^0.8.17 (contracts/mosaic-alpha-contracts/User/Affiliate.sol#2) necessitates a version too recent to be trusted. Consider deploying with
0.6.12/0.7.6/0.8.7
Pragma version^0.8.17 (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#2) necessitates a version too recent to be trusted. Consider deploying
with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.17 (contracts/mosaic-alpha-contracts/User/Affiliate_old.sol#2) necessitates a version too recent to be trusted. Consider deploying with
0.6.12/0.7.6/0.8.7
Pragma version^0.8.17 (contracts/mosaic-alpha-contracts/User/SimpleAffiliateBoosterController.sol#2) necessitates a version too recent to be trusted.
Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.17 (contracts/mosaic-alpha-contracts/User/UserProfile.sol#2) necessitates a version too recent to be trusted. Consider deploying with
0.6.12/0.7.6/0.8.7
Pragma version^0.8.17 (contracts/mosaic-alpha-contracts/Vault/FlashLoans.sol#2) necessitates a version too recent to be trusted. Consider deploying with
0.6.12/0.7.6/0.8.7
Pragma version^0.8.17 (contracts/mosaic-alpha-contracts/Vault/Swaps.sol#2) necessitates a version too recent to be trusted. Consider deploying with
0.6.12/0.7.6/0.8.7
Pragma version^0.8.17 (contracts/mosaic-alpha-contracts/helpers/IEntropy.sol#2) necessitates a version too recent to be trusted. Consider deploying with
0.6.12/0.7.6/0.8.7
Pragma version^0.8.17 (contracts/mosaic-alpha-contracts/helpers/ISystemRole.sol#2) necessitates a version too recent to be trusted. Consider deploying with
0.6.12/0.7.6/0.8.7
Pragma version^0.8.17 (contracts/mosaic-alpha-contracts/helpers/ISystemWhitelist.sol#2) necessitates a version too recent to be trusted. Consider deploying
with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.17 (contracts/mosaic-alpha-contracts/helpers/SimpleEntropy.sol#3) necessitates a version too recent to be trusted. Consider deploying
with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.17 (contracts/mosaic-alpha-contracts/helpers/SystemRole.sol#2) necessitates a version too recent to be trusted. Consider deploying with
0.6.12/0.7.6/0.8.7
Pragma version^0.8.17 (contracts/mosaic-alpha-contracts/helpers/SystemWhitelist.sol#2) necessitates a version too recent to be trusted. Consider deploying
with 0.6.12/0.7.6/0.8.7
Pragma version>=0.5.0<0.9.0 (contracts/mosaic-alpha-contracts/helpers/pancake/interfaces/IPancakeOracle.sol#2) is too complex
Pragma version>=0.5.0 (contracts/mosaic-alpha-contracts/helpers/pancake/libraries/Babylonian.sol#3) allows old versions
Pragma version>=0.5.0 (contracts/mosaic-alpha-contracts/helpers/pancake/libraries/BitMath.sol#2) allows old versions
Pragma version>=0.5.0 (contracts/mosaic-alpha-contracts/helpers/pancake/libraries/FixedPoint.sol#2) allows old versions
Pragma version>=0.5.0 (contracts/mosaic-alpha-contracts/helpers/pancake/libraries/FullMath.sol#2) allows old versions
Pragma version>=0.5.0<7.0.0 (contracts/mosaic-alpha-contracts/helpers/pancake/libraries/PancakeOracleLibrary.sol#2) is too complex
Pragma version^0.8.17 (contracts/test/ConfigGovernedExample.sol#2) necessitates a version too recent to be trusted. Consider deploying with
0.6.12/0.7.6/0.8.7
Pragma version^0.8.17 (contracts/test/TestToken.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
```

low-level-calls

```
Low level call in PancakePair._safeTransfer(address,address,uint256) (contracts/mosaic-alpha-contracts/helpers/pancake/PancakePair.sol#46-49):
    - (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value)) (contracts/mosaic-alpha-contracts/helpers/pancake/PancakePair.sol#47)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Low level call in TransferHelper.safeApprove(address,address,uint256) (contracts/mosaic-alpha-contracts/helpers/pancake/libraries/TransferHelper.sol#6-10):
    - (success,data) = token.call(abi.encodeWithSelector(0x095ea7b3,to,value)) (contracts/mosaic-alpha-
contracts/helpers/pancake/libraries/TransferHelper.sol#8)
Low level call in TransferHelper.safeTransfer(address,address,uint256) (contracts/mosaic-alpha-contracts/helpers/pancake/libraries/TransferHelper.sol#12-
16):
    - (success,data) = token.call(abi.encodeWithSelector(0xa9059cbb,to,value)) (contracts/mosaic-alpha-
contracts/helpers/pancake/libraries/TransferHelper.sol#14)
Low level call in TransferHelper.safeTransferFrom(address,address,address,uint256) (contracts/mosaic-alpha-
contracts/helpers/pancake/libraries/TransferHelper.sol#18-22):
    - (success,data) = token.call(abi.encodeWithSelector(0x23b872dd,from,to,value)) (contracts/mosaic-alpha-
contracts/helpers/pancake/libraries/TransferHelper.sol#20)
Low level call in TransferHelper.safeTransferETH(address,uint256) (contracts/mosaic-alpha-contracts/helpers/pancake/libraries/TransferHelper.sol#24-27):
    - (success) = to.call{value: value}(new bytes(0)) (contracts/mosaic-alpha-contracts/helpers/pancake/libraries/TransferHelper.sol#25)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Low level call in Address.sendValue(address,uint256) (node_modules/@openzeppelin/contracts/utils/Address.sol#60-65):
    - (success) = recipient.call{value: amount}() (node_modules/@openzeppelin/contracts/utils/Address.sol#63)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#128-139):
    - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#137)
Low level call in Address.functionStaticCall(address,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#157-166):
    - (success,returndata) = target.staticcall(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#164)
Low level call in Address.functionDelegateCall(address,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#184-193):
    - (success,returndata) = target.delegatecall(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#191)
Low level call in Governance._execute_ManageBytes(address,string,bytes) (contracts/mosaic-alpha-contracts/Governance/Governance.sol#507-524):
    - (success,retData) = _contractA.call(abi.encodePacked(signature,_data)) (contracts/mosaic-alpha-contracts/Governance/Governance.sol#522)
Low level call in Governance.execute_ManageList(address,string,address[]) (contracts/mosaic-alpha-contracts/Governance/Governance.sol#527-531):
    - (success,retData) = _contractA.call(abi.encodeWithSignature(_funcName,address_array)) (contracts/mosaic-alpha-
contracts/Governance/Governance.sol#529)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Low level call in Address.sendValue(address,uint256) (contracts/.deps/npm/@openzeppelin/contracts/utils/Address.sol#60-65):
    - (success) = recipient.call{value: amount}() (contracts/.deps/npm/@openzeppelin/contracts/utils/Address.sol#63)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (contracts/.deps/npm/@openzeppelin/contracts/utils/Address.sol#128-137):
    - (success,returndata) = target.call{value: value}(data) (contracts/.deps/npm/@openzeppelin/contracts/utils/Address.sol#135)
Low level call in Address.functionStaticCall(address,bytes,string) (contracts/.deps/npm/@openzeppelin/contracts/utils/Address.sol#155-162):
```

```
    - (success,returndata) = target.staticcall(data) (contracts/.deps/npm/@openzeppelin/contracts/utils/Address.sol#160)
Low level call in Address.functionDelegateCall(address,bytes,string) (contracts/.deps/npm/@openzeppelin/contracts/utils/Address.sol#180-187):
    - (success,returndata) = target.delegatecall(data) (contracts/.deps/npm/@openzeppelin/contracts/utils/Address.sol#185)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
```

missing-inheritance

```
GovernedContractsMM (contracts/test/ConfigGovernedExample.sol#10-60) should inherit from IGoverned (contracts/mosaic-alpha-
contracts/Interfaces/IGoverned.sol#4-12)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-inheritance
```

conformance-to-solidity-naming-conventions

```
Variable PancakeERC20.DOMAIN_SEPARATOR (contracts/mosaic-alpha-contracts/helpers/pancake/PancakeERC20.sol#18) is not in mixedCase
Parameter PancakeFactory.setFeeTo(address)._feeTo (contracts/mosaic-alpha-contracts/helpers/pancake/PancakeFactory.sol#46) is not in mixedCase
Parameter PancakeFactory.setFeeToSetter(address)._feeToSetter (contracts/mosaic-alpha-contracts/helpers/pancake/PancakeFactory.sol#51) is not in mixedCase
Parameter PancakePair.initialize(address,address)._token0 (contracts/mosaic-alpha-contracts/helpers/pancake/PancakePair.sol#68) is not in mixedCase
Parameter PancakePair.initialize(address,address)._token1 (contracts/mosaic-alpha-contracts/helpers/pancake/PancakePair.sol#68) is not in mixedCase
Function IPancakeERC20.DOMAIN_SEPARATOR() (contracts/mosaic-alpha-contracts/helpers/pancake/interfaces/IPancakeERC20.sol#20) is not in mixedCase
Function IPancakeERC20.PERMIT_TYPEHASH() (contracts/mosaic-alpha-contracts/helpers/pancake/interfaces/IPancakeERC20.sol#21) is not in mixedCase
Function IPancakePair.DOMAIN_SEPARATOR() (contracts/mosaic-alpha-contracts/helpers/pancake/interfaces/IPancakePair.sol#20) is not in mixedCase
Function IPancakePair.PERMIT_TYPEHASH() (contracts/mosaic-alpha-contracts/helpers/pancake/interfaces/IPancakePair.sol#21) is not in mixedCase
Function IPancakePair.MINIMUM_LIQUIDITY() (contracts/mosaic-alpha-contracts/helpers/pancake/interfaces/IPancakePair.sol#38) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Variable PancakeRouter.WETH (contracts/mosaic-alpha-contracts/helpers/pancake/PancakeRouter.sol#18) is not in mixedCase
Function IPancakeRouter01.WETH() (contracts/mosaic-alpha-contracts/helpers/pancake/interfaces/IPancakeRouter01.sol#8) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Function IPool.VERSION() (contracts/mosaic-alpha-contracts/Interfaces/IPool.sol#9) is not in mixedCase
Function IPool._mint(uint256,IVault.TotalSupplyBase) (contracts/mosaic-alpha-contracts/Interfaces/IPool.sol#98) is not in mixedCase
Parameter IPool._mint(uint256,IVault.TotalSupplyBase).LPTokens (contracts/mosaic-alpha-contracts/Interfaces/IPool.sol#98) is not in mixedCase
Function IPool._burn(uint256,IVault.TotalSupplyBase) (contracts/mosaic-alpha-contracts/Interfaces/IPool.sol#101) is not in mixedCase
Parameter IPool._burn(uint256,IVault.TotalSupplyBase).LPTokens (contracts/mosaic-alpha-contracts/Interfaces/IPool.sol#101) is not in mixedCase
Parameter IPool.calcMintAmounts(uint256).LPTokensToMint (contracts/mosaic-alpha-contracts/Interfaces/IPool.sol#104) is not in mixedCase
Parameter IPool.calcBurnAmounts(uint256).LPTokensToBurn (contracts/mosaic-alpha-contracts/Interfaces/IPool.sol#107) is not in mixedCase
Function IVault.AddRemoveAdmin(address,bool) (contracts/mosaic-alpha-contracts/Interfaces/IVault.sol#61) is not in mixedCase
Parameter IVault.AddRemoveAdmin(address,bool)._ShouldBeAdmin (contracts/mosaic-alpha-contracts/Interfaces/IVault.sol#61) is not in mixedCase
Function IVault.AddRemoveBoostedPool(address,bool) (contracts/mosaic-alpha-contracts/Interfaces/IVault.sol#66) is not in mixedCase
Parameter IVault.AddRemoveBoostedPool(address,bool)._ShouldBeBoosted (contracts/mosaic-alpha-contracts/Interfaces/IVault.sol#66) is not in mixedCase
Function IVault.Mint(uint32,uint256,uint256[],address,address,bool,uint256,uint256) (contracts/mosaic-alpha-contracts/Interfaces/IVault.sol#201) is not in
mixedCase
Parameter IVault.Mint(uint32,uint256,uint256[],address,address,bool,uint256,uint256).LPTokensRequested (contracts/mosaic-alpha-
contracts/Interfaces/IVault.sol#201) is not in mixedCase
Function IVault.Burn(uint32,uint256,uint256[],bool,uint256,address) (contracts/mosaic-alpha-contracts/Interfaces/IVault.sol#207) is not in mixedCase
Parameter IVault.Burn(uint32,uint256,uint256[],bool,uint256,address).LPTokensToBurn (contracts/mosaic-alpha-contracts/Interfaces/IVault.sol#207) is not in
mixedCase
Function FixedPoint._ln_36(int256) (contracts/mosaic-alpha-contracts/Pool/Math.sol#561-608) is not in mixedCase
Constant FixedPoint.x0 (contracts/mosaic-alpha-contracts/Pool/Math.sol#156) is not in UPPER_CASE_WITH_UNDERSCORES
Constant FixedPoint.a0 (contracts/mosaic-alpha-contracts/Pool/Math.sol#157) is not in UPPER_CASE_WITH_UNDERSCORES
Constant FixedPoint.x1 (contracts/mosaic-alpha-contracts/Pool/Math.sol#158) is not in UPPER_CASE_WITH_UNDERSCORES
Constant FixedPoint.a1 (contracts/mosaic-alpha-contracts/Pool/Math.sol#159) is not in UPPER_CASE_WITH_UNDERSCORES
Constant FixedPoint.x2 (contracts/mosaic-alpha-contracts/Pool/Math.sol#162) is not in UPPER_CASE_WITH_UNDERSCORES
Constant FixedPoint.a2 (contracts/mosaic-alpha-contracts/Pool/Math.sol#163) is not in UPPER_CASE_WITH_UNDERSCORES
Constant FixedPoint.x3 (contracts/mosaic-alpha-contracts/Pool/Math.sol#164) is not in UPPER_CASE_WITH_UNDERSCORES
Constant FixedPoint.a3 (contracts/mosaic-alpha-contracts/Pool/Math.sol#165) is not in UPPER_CASE_WITH_UNDERSCORES
Constant FixedPoint.x4 (contracts/mosaic-alpha-contracts/Pool/Math.sol#166) is not in UPPER_CASE_WITH_UNDERSCORES
Constant FixedPoint.a4 (contracts/mosaic-alpha-contracts/Pool/Math.sol#167) is not in UPPER_CASE_WITH_UNDERSCORES
Constant FixedPoint.x5 (contracts/mosaic-alpha-contracts/Pool/Math.sol#168) is not in UPPER_CASE_WITH_UNDERSCORES
Constant FixedPoint.a5 (contracts/mosaic-alpha-contracts/Pool/Math.sol#169) is not in UPPER_CASE_WITH_UNDERSCORES
Constant FixedPoint.x6 (contracts/mosaic-alpha-contracts/Pool/Math.sol#170) is not in UPPER_CASE_WITH_UNDERSCORES
Constant FixedPoint.a6 (contracts/mosaic-alpha-contracts/Pool/Math.sol#171) is not in UPPER_CASE_WITH_UNDERSCORES
Constant FixedPoint.x7 (contracts/mosaic-alpha-contracts/Pool/Math.sol#172) is not in UPPER_CASE_WITH_UNDERSCORES
Constant FixedPoint.a7 (contracts/mosaic-alpha-contracts/Pool/Math.sol#173) is not in UPPER_CASE_WITH_UNDERSCORES
Constant FixedPoint.x8 (contracts/mosaic-alpha-contracts/Pool/Math.sol#174) is not in UPPER_CASE_WITH_UNDERSCORES
Constant FixedPoint.a8 (contracts/mosaic-alpha-contracts/Pool/Math.sol#175) is not in UPPER_CASE_WITH_UNDERSCORES
Constant FixedPoint.x9 (contracts/mosaic-alpha-contracts/Pool/Math.sol#176) is not in UPPER_CASE_WITH_UNDERSCORES
Constant FixedPoint.a9 (contracts/mosaic-alpha-contracts/Pool/Math.sol#177) is not in UPPER_CASE_WITH_UNDERSCORES
Constant FixedPoint.x10 (contracts/mosaic-alpha-contracts/Pool/Math.sol#178) is not in UPPER_CASE_WITH_UNDERSCORES
Constant FixedPoint.a10 (contracts/mosaic-alpha-contracts/Pool/Math.sol#179) is not in UPPER_CASE_WITH_UNDERSCORES
Constant FixedPoint.x11 (contracts/mosaic-alpha-contracts/Pool/Math.sol#180) is not in UPPER_CASE_WITH_UNDERSCORES
Constant FixedPoint.a11 (contracts/mosaic-alpha-contracts/Pool/Math.sol#181) is not in UPPER_CASE_WITH_UNDERSCORES
Function Pool.VERSION() (contracts/mosaic-alpha-contracts/Pool/Pool.sol#57-59) is not in mixedCase
Function Pool.Live() (contracts/mosaic-alpha-contracts/Pool/Pool.sol#107-109) is not in mixedCase
Parameter Pool.initialize(string,string,uint8,address,address,address[],uint32[],uint256,IFees.MosaicPoolFees,address,bool)._name (contracts/mosaic-alpha-
contracts/Pool/Pool.sol#172) is not in mixedCase
Parameter Pool.initialize(string,string,uint8,address,address,address[],uint32[],uint256,IFees.MosaicPoolFees,address,bool)._symbol (contracts/mosaic-
alpha-contracts/Pool/Pool.sol#173) is not in mixedCase
Parameter Pool.initialize(string,string,uint8,address,address,address[],uint32[],uint256,IFees.MosaicPoolFees,address,bool)._poolType (contracts/mosaic-
alpha-contracts/Pool/Pool.sol#174) is not in mixedCase
Parameter Pool.initialize(string,string,uint8,address,address,address[],uint32[],uint256,IFees.MosaicPoolFees,address,bool)._vaultAddress
(contracts/mosaic-alpha-contracts/Pool/Pool.sol#175) is not in mixedCase
Parameter Pool.initialize(string,string,uint8,address,address,address[],uint32[],uint256,IFees.MosaicPoolFees,address,bool)._feesAddress (contracts/mosaic-
alpha-contracts/Pool/Pool.sol#176) is not in mixedCase
Parameter Pool.initialize(string,string,uint8,address,address,address[],uint32[],uint256,IFees.MosaicPoolFees,address,bool)._tokens (contracts/mosaic-
alpha-contracts/Pool/Pool.sol#177) is not in mixedCase
Parameter Pool.initialize(string,string,uint8,address,address,address[],uint32[],uint256,IFees.MosaicPoolFees,address,bool)._weights (contracts/mosaic-
alpha-contracts/Pool/Pool.sol#178) is not in mixedCase
Parameter Pool.initialize(string,string,uint8,address,address,address[],uint32[],uint256,IFees.MosaicPoolFees,address,bool)._lpTokens (contracts/mosaic-
alpha-contracts/Pool/Pool.sol#180) is not in mixedCase
Parameter Pool.initialize(string,string,uint8,address,address,address[],uint32[],uint256,IFees.MosaicPoolFees,address,bool)._poolFees (contracts/mosaic-
alpha-contracts/Pool/Pool.sol#181) is not in mixedCase
Parameter Pool.initialize(string,string,uint8,address,address,address[],uint32[],uint256,IFees.MosaicPoolFees,address,bool)._owner (contracts/mosaic-alpha-
contracts/Pool/Pool.sol#182) is not in mixedCase
Parameter Pool.initialize(string,string,uint8,address,address,address[],uint32[],uint256,IFees.MosaicPoolFees,address,bool)._feeless (contracts/mosaic-
alpha-contracts/Pool/Pool.sol#183) is not in mixedCase
Parameter Pool.setManagers(address[])._managers (contracts/mosaic-alpha-contracts/Pool/Pool.sol#268) is not in mixedCase
Parameter Pool.balanceOf(address)._owner (contracts/mosaic-alpha-contracts/Pool/Pool.sol#277) is not in mixedCase
Parameter Pool.transfer(address,uint256)._to (contracts/mosaic-alpha-contracts/Pool/Pool.sol#281) is not in mixedCase
Parameter Pool.transfer(address,uint256)._value (contracts/mosaic-alpha-contracts/Pool/Pool.sol#281) is not in mixedCase
Parameter Pool.safeTransfer(address,uint256)._to (contracts/mosaic-alpha-contracts/Pool/Pool.sol#285) is not in mixedCase
Parameter Pool.safeTransfer(address,uint256)._value (contracts/mosaic-alpha-contracts/Pool/Pool.sol#285) is not in mixedCase
Parameter Pool.transferFrom(address,address,uint256)._from (contracts/mosaic-alpha-contracts/Pool/Pool.sol#295) is not in mixedCase
Parameter Pool.transferFrom(address,address,uint256)._to (contracts/mosaic-alpha-contracts/Pool/Pool.sol#295) is not in mixedCase
Parameter Pool.transferFrom(address,address,uint256)._value (contracts/mosaic-alpha-contracts/Pool/Pool.sol#295) is not in mixedCase
Parameter Pool.safeTransferFrom(address,address,uint256)._from (contracts/mosaic-alpha-contracts/Pool/Pool.sol#299) is not in mixedCase
Parameter Pool.safeTransferFrom(address,address,uint256)._to (contracts/mosaic-alpha-contracts/Pool/Pool.sol#299) is not in mixedCase
Parameter Pool.safeTransferFrom(address,address,uint256)._value (contracts/mosaic-alpha-contracts/Pool/Pool.sol#299) is not in mixedCase
Parameter Pool.emitTransfer(address,address,uint256)._from (contracts/mosaic-alpha-contracts/Pool/Pool.sol#313) is not in mixedCase
Parameter Pool.emitTransfer(address,address,uint256)._to (contracts/mosaic-alpha-contracts/Pool/Pool.sol#313) is not in mixedCase
Parameter Pool.emitTransfer(address,address,uint256)._value (contracts/mosaic-alpha-contracts/Pool/Pool.sol#313) is not in mixedCase
Parameter Pool.approve(address,uint256)._spender (contracts/mosaic-alpha-contracts/Pool/Pool.sol#319) is not in mixedCase
Parameter Pool.approve(address,uint256)._value (contracts/mosaic-alpha-contracts/Pool/Pool.sol#319) is not in mixedCase
Parameter Pool.allowance(address,address)._owner (contracts/mosaic-alpha-contracts/Pool/Pool.sol#326) is not in mixedCase
```

```
Parameter Pool.allowance(address,address)._spender (contracts/mosaic-alpha-contracts/Pool/Pool.sol#326) is not in mixedCase
Parameter Pool.calcMintAmounts(uint256).LPTokensToMint (contracts/mosaic-alpha-contracts/Pool/Pool.sol#334) is not in mixedCase
Parameter Pool.calcBurnAmounts(uint256).LPTokensToBurn (contracts/mosaic-alpha-contracts/Pool/Pool.sol#350) is not in mixedCase
Function Pool._mint(uint256,IVault.TotalSupplyBase) (contracts/mosaic-alpha-contracts/Pool/Pool.sol#373-383) is not in mixedCase
Parameter Pool._mint(uint256,IVault.TotalSupplyBase).LPTokens (contracts/mosaic-alpha-contracts/Pool/Pool.sol#373) is not in mixedCase
Function Pool._burn(uint256,IVault.TotalSupplyBase) (contracts/mosaic-alpha-contracts/Pool/Pool.sol#386-395) is not in mixedCase
Parameter Pool._burn(uint256,IVault.TotalSupplyBase).LPTokens (contracts/mosaic-alpha-contracts/Pool/Pool.sol#386) is not in mixedCase
Function Pool._calcOutGivenIn(uint256,uint256,uint256,uint256,uint256,uint16) (contracts/mosaic-alpha-contracts/Pool/Pool.sol#401-438) is not in mixedCase
Function Pool._calcInGivenOut(uint256,uint256,uint256,uint256,uint256,uint16) (contracts/mosaic-alpha-contracts/Pool/Pool.sol#444-484) is not in mixedCase
Parameter Pool.addToken(address)._token (contracts/mosaic-alpha-contracts/Pool/Pool.sol#535) is not in mixedCase
Parameter Pool.removeToken(uint256)._index (contracts/mosaic-alpha-contracts/Pool/Pool.sol#569) is not in mixedCase
Parameter Pool.updateWeights(uint32,uint32[])._newWeights (contracts/mosaic-alpha-contracts/Pool/Pool.sol#667) is not in mixedCase
Parameter Pool.calcWeight(uint32,uint32,uint32,uint32)._weight (contracts/mosaic-alpha-contracts/Pool/Pool.sol#685) is not in mixedCase
Parameter Pool.calcWeight(uint32,uint32,uint32,uint32)._newWeight (contracts/mosaic-alpha-contracts/Pool/Pool.sol#685) is not in mixedCase
Parameter Pool.calcWeight(uint32,uint32,uint32,uint32)._changeStart (contracts/mosaic-alpha-contracts/Pool/Pool.sol#685) is not in mixedCase
Parameter Pool.calcWeight(uint32,uint32,uint32,uint32)._changeEnd (contracts/mosaic-alpha-contracts/Pool/Pool.sol#685) is not in mixedCase
Variable Pool.MIN_DUST_AMOUNT (contracts/mosaic-alpha-contracts/Pool/Pool.sol#20) is not in mixedCase
Variable Pool._MAX_INOUT_RATIO (contracts/mosaic-alpha-contracts/Pool/Pool.sol#24) is not in mixedCase
Variable Pool._MIN_WEIGHTUPDATE_DURATION (contracts/mosaic-alpha-contracts/Pool/Pool.sol#25) is not in mixedCase
Contract console (node_modules/hardhat/console.sol#4-1532) is not in CapWords
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Function IERC20Permit.DOMAIN_SEPARATOR() (node_modules/@openzeppelin/contracts/token/ERC20/extensions/draft-IERC20Permit.sol#59) is not in mixedCase
Struct Governance.config_struct (contracts/mosaic-alpha-contracts/Governance/Governance.sol#28-37) is not in CapWords
Event GovernanceTransfer_Proportion(uint256) (contracts/mosaic-alpha-contracts/Governance/Governance.sol#57) is not in CapWords
Event GovernanceAction_Proposed(uint256) (contracts/mosaic-alpha-contracts/Governance/Governance.sol#58) is not in CapWords
Event GovernanceAction_Support(uint256) (contracts/mosaic-alpha-contracts/Governance/Governance.sol#59) is not in CapWords
Event GovernanceAction_Trigger(uint256) (contracts/mosaic-alpha-contracts/Governance/Governance.sol#60) is not in CapWords
Event GovernanceConfig_Proposed(string) (contracts/mosaic-alpha-contracts/Governance/Governance.sol#61) is not in CapWords
Event GovernanceConfig_Supported(string) (contracts/mosaic-alpha-contracts/Governance/Governance.sol#62) is not in CapWords
Function Governance.core_govAddr_conf(address) (contracts/mosaic-alpha-contracts/Governance/Governance.sol#97-99) is not in mixedCase
Parameter Governance.core_govAddr_conf(address)._address (contracts/mosaic-alpha-contracts/Governance/Governance.sol#97) is not in mixedCase
Function Governance.core_emergency_conf() (contracts/mosaic-alpha-contracts/Governance/Governance.sol#101-102) is not in mixedCase
Function Governance.core_managers_conf(address[]) (contracts/mosaic-alpha-contracts/Governance/Governance.sol#104-109) is not in mixedCase
Parameter Governance.core_managers_conf(address[])._addresses (contracts/mosaic-alpha-contracts/Governance/Governance.sol#104) is not in mixedCase
Function Governance.ActivateDeployedMosaic(address,address,address,address,address,address,address,address,address,address,address) (contracts/mosaic-
alpha-contracts/Governance/Governance.sol#128-171) is not in mixedCase
Parameter Governance.ActivateDeployedMosaic(address,address,address,address,address,address,address,address,address,address,address)._userProfile
(contracts/mosaic-alpha-contracts/Governance/Governance.sol#129) is not in mixedCase
Parameter Governance.ActivateDeployedMosaic(address,address,address,address,address,address,address,address,address,address,address)._affiliate
(contracts/mosaic-alpha-contracts/Governance/Governance.sol#130) is not in mixedCase
Parameter Governance.ActivateDeployedMosaic(address,address,address,address,address,address,address,address,address,address,address)._fees
(contracts/mosaic-alpha-contracts/Governance/Governance.sol#131) is not in mixedCase
Parameter Governance.ActivateDeployedMosaic(address,address,address,address,address,address,address,address,address,address,address)._register
(contracts/mosaic-alpha-contracts/Governance/Governance.sol#132) is not in mixedCase
Parameter Governance.ActivateDeployedMosaic(address,address,address,address,address,address,address,address,address,address,address)._poolFactory
(contracts/mosaic-alpha-contracts/Governance/Governance.sol#133) is not in mixedCase
Parameter Governance.ActivateDeployedMosaic(address,address,address,address,address,address,address,address,address,address,address)._feeTo
(contracts/mosaic-alpha-contracts/Governance/Governance.sol#134) is not in mixedCase
Parameter Governance.ActivateDeployedMosaic(address,address,address,address,address,address,address,address,address,address,address)._swapsContract
(contracts/mosaic-alpha-contracts/Governance/Governance.sol#135) is not in mixedCase
Parameter Governance.ActivateDeployedMosaic(address,address,address,address,address,address,address,address,address,address,address)._oracle
(contracts/mosaic-alpha-contracts/Governance/Governance.sol#136) is not in mixedCase
Parameter Governance.ActivateDeployedMosaic(address,address,address,address,address,address,address,address,address,address,address)._deposit
(contracts/mosaic-alpha-contracts/Governance/Governance.sol#137) is not in mixedCase
Parameter Governance.ActivateDeployedMosaic(address,address,address,address,address,address,address,address,address,address,address)._burner
(contracts/mosaic-alpha-contracts/Governance/Governance.sol#138) is not in mixedCase
Parameter Governance.ActivateDeployedMosaic(address,address,address,address,address,address,address,address,address,address,address)._booster
(contracts/mosaic-alpha-contracts/Governance/Governance.sol#139) is not in mixedCase
Function Governance.transfer_proportion(address,uint256) (contracts/mosaic-alpha-contracts/Governance/Governance.sol#174-182) is not in mixedCase
Parameter Governance.transfer_proportion(address,uint256)._address (contracts/mosaic-alpha-contracts/Governance/Governance.sol#174) is not in mixedCase
Parameter Governance.transfer_proportion(address,uint256)._amount (contracts/mosaic-alpha-contracts/Governance/Governance.sol#174) is not in mixedCase
Function Governance.update_config(string,bool,address,address[],bool[],address[],uint256[],bytes32[]) (contracts/mosaic-alpha-
contracts/Governance/Governance.sol#187-205) is not in mixedCase
Parameter Governance.update_config(string,bool,address,address[],bool[],address[],uint256[],bytes32[])._name (contracts/mosaic-alpha-
contracts/Governance/Governance.sol#187) is not in mixedCase
Parameter Governance.update_config(string,bool,address,address[],bool[],address[],uint256[],bytes32[])._Running (contracts/mosaic-alpha-
contracts/Governance/Governance.sol#188) is not in mixedCase
Parameter Governance.update_config(string,bool,address,address[],bool[],address[],uint256[],bytes32[])._govaddr (contracts/mosaic-alpha-
contracts/Governance/Governance.sol#189) is not in mixedCase
Parameter Governance.update_config(string,bool,address,address[],bool[],address[],uint256[],bytes32[])._managers (contracts/mosaic-alpha-
contracts/Governance/Governance.sol#190) is not in mixedCase
Parameter Governance.update_config(string,bool,address,address[],bool[],address[],uint256[],bytes32[])._boolslot (contracts/mosaic-alpha-
contracts/Governance/Governance.sol#191) is not in mixedCase
Parameter Governance.update_config(string,bool,address,address[],bool[],address[],uint256[],bytes32[])._address_slot (contracts/mosaic-alpha-
contracts/Governance/Governance.sol#192) is not in mixedCase
Parameter Governance.update_config(string,bool,address,address[],bool[],address[],uint256[],bytes32[])._uint256_slot (contracts/mosaic-alpha-
contracts/Governance/Governance.sol#193) is not in mixedCase
Parameter Governance.update_config(string,bool,address,address[],bool[],address[],uint256[],bytes32[])._bytes32_slot (contracts/mosaic-alpha-
contracts/Governance/Governance.sol#194) is not in mixedCase
Function Governance.votein_config(string,bool,address,address[],bool[],address[],uint256[],bytes32[]) (contracts/mosaic-alpha-
contracts/Governance/Governance.sol#208-226) is not in mixedCase
Parameter Governance.votein_config(string,bool,address,address[],bool[],address[],uint256[],bytes32[])._name (contracts/mosaic-alpha-
contracts/Governance/Governance.sol#208) is not in mixedCase
Parameter Governance.votein_config(string,bool,address,address[],bool[],address[],uint256[],bytes32[])._Running (contracts/mosaic-alpha-
contracts/Governance/Governance.sol#209) is not in mixedCase
Parameter Governance.votein_config(string,bool,address,address[],bool[],address[],uint256[],bytes32[])._govaddr (contracts/mosaic-alpha-
contracts/Governance/Governance.sol#210) is not in mixedCase
Parameter Governance.votein_config(string,bool,address,address[],bool[],address[],uint256[],bytes32[])._managers (contracts/mosaic-alpha-
contracts/Governance/Governance.sol#211) is not in mixedCase
Parameter Governance.votein_config(string,bool,address,address[],bool[],address[],uint256[],bytes32[])._boolslot (contracts/mosaic-alpha-
contracts/Governance/Governance.sol#212) is not in mixedCase
Parameter Governance.votein_config(string,bool,address,address[],bool[],address[],uint256[],bytes32[])._address_slot (contracts/mosaic-alpha-
contracts/Governance/Governance.sol#213) is not in mixedCase
Parameter Governance.votein_config(string,bool,address,address[],bool[],address[],uint256[],bytes32[])._uint256_slot (contracts/mosaic-alpha-
contracts/Governance/Governance.sol#214) is not in mixedCase
Parameter Governance.votein_config(string,bool,address,address[],bool[],address[],uint256[],bytes32[])._bytes32_slot (contracts/mosaic-alpha-
contracts/Governance/Governance.sol#215) is not in mixedCase
Function Governance.propose_config(string,bool,address,address[],bool[],address[],uint256[],bytes32[]) (contracts/mosaic-alpha-
contracts/Governance/Governance.sol#229-260) is not in mixedCase
Parameter Governance.propose_config(string,bool,address,address[],bool[],address[],uint256[],bytes32[])._name (contracts/mosaic-alpha-
contracts/Governance/Governance.sol#230) is not in mixedCase
Parameter Governance.propose_config(string,bool,address,address[],bool[],address[],uint256[],bytes32[])._Running (contracts/mosaic-alpha-
contracts/Governance/Governance.sol#231) is not in mixedCase
Parameter Governance.propose_config(string,bool,address,address[],bool[],address[],uint256[],bytes32[])._govaddr (contracts/mosaic-alpha-
contracts/Governance/Governance.sol#232) is not in mixedCase
Parameter Governance.propose_config(string,bool,address,address[],bool[],address[],uint256[],bytes32[])._managers (contracts/mosaic-alpha-
contracts/Governance/Governance.sol#233) is not in mixedCase
Parameter Governance.propose_config(string,bool,address,address[],bool[],address[],uint256[],bytes32[])._boolslot (contracts/mosaic-alpha-
contracts/Governance/Governance.sol#234) is not in mixedCase
Parameter Governance.propose_config(string,bool,address,address[],bool[],address[],uint256[],bytes32[])._address_slot (contracts/mosaic-alpha-
contracts/Governance/Governance.sol#235) is not in mixedCase
Parameter Governance.propose_config(string,bool,address,address[],bool[],address[],uint256[],bytes32[])._uint256_slot (contracts/mosaic-alpha-
contracts/Governance/Governance.sol#236) is not in mixedCase
Parameter Governance.propose_config(string,bool,address,address[],bool[],address[],uint256[],bytes32[])._bytes32_slot (contracts/mosaic-alpha-
contracts/Governance/Governance.sol#237) is not in mixedCase
Function Governance.propose_core_change(address,bool,address[],address[]) (contracts/mosaic-alpha-contracts/Governance/Governance.sol#263-279) is not in
```

mixedCase
Parameter Governance.propose_core_change(address,bool,address[],address[])._govaddr (contracts/mosaic-alpha-contracts/Governance/Governance.sol#263) is not in mixedCase
Parameter Governance.propose_core_change(address,bool,address[],address[])._Running (contracts/mosaic-alpha-contracts/Governance/Governance.sol#263) is not in mixedCase
Parameter Governance.propose_core_change(address,bool,address[],address[])._managers (contracts/mosaic-alpha-contracts/Governance/Governance.sol#263) is not in mixedCase
Parameter Governance.propose_core_change(address,bool,address[],address[])._owners (contracts/mosaic-alpha-contracts/Governance/Governance.sol#263) is not in mixedCase
Function Governance.support_config_proposal(uint256,string) (contracts/mosaic-alpha-contracts/Governance/Governance.sol#282-317) is not in mixedCase
Parameter Governance.support_config_proposal(uint256,string)._confCount (contracts/mosaic-alpha-contracts/Governance/Governance.sol#282) is not in mixedCase
Parameter Governance.support_config_proposal(uint256,string)._name (contracts/mosaic-alpha-contracts/Governance/Governance.sol#282) is not in mixedCase
Function Governance.read_core_Running() (contracts/mosaic-alpha-contracts/Governance/Governance.sol#321) is not in mixedCase
Function Governance.read_core_govAddr() (contracts/mosaic-alpha-contracts/Governance/Governance.sol#322) is not in mixedCase
Function Governance.read_core_managers() (contracts/mosaic-alpha-contracts/Governance/Governance.sol#323) is not in mixedCase
Function Governance.read_core_owners() (contracts/mosaic-alpha-contracts/Governance/Governance.sol#324) is not in mixedCase
Function Governance.read_config_Main_addressN(uint256) (contracts/mosaic-alpha-contracts/Governance/Governance.sol#326-328) is not in mixedCase
Parameter Governance.read_config_Main_addressN(uint256)._n (contracts/mosaic-alpha-contracts/Governance/Governance.sol#326) is not in mixedCase
Function Governance.read_config_core(string) (contracts/mosaic-alpha-contracts/Governance/Governance.sol#331-340) is not in mixedCase
Parameter Governance.read_config_core(string)._name (contracts/mosaic-alpha-contracts/Governance/Governance.sol#331) is not in mixedCase
Function Governance.read_config_name(string) (contracts/mosaic-alpha-contracts/Governance/Governance.sol#341) is not in mixedCase
Parameter Governance.read_config_name(string)._name (contracts/mosaic-alpha-contracts/Governance/Governance.sol#341) is not in mixedCase
Function Governance.read_config_emergencyStatus(string) (contracts/mosaic-alpha-contracts/Governance/Governance.sol#342) is not in mixedCase
Parameter Governance.read_config_emergencyStatus(string)._name (contracts/mosaic-alpha-contracts/Governance/Governance.sol#342) is not in mixedCase
Function Governance.read_config_governAddress(string) (contracts/mosaic-alpha-contracts/Governance/Governance.sol#343) is not in mixedCase
Parameter Governance.read_config_governAddress(string)._name (contracts/mosaic-alpha-contracts/Governance/Governance.sol#343) is not in mixedCase
Function Governance.read_config_Managers(string) (contracts/mosaic-alpha-contracts/Governance/Governance.sol#344) is not in mixedCase
Parameter Governance.read_config_Managers(string)._name (contracts/mosaic-alpha-contracts/Governance/Governance.sol#344) is not in mixedCase
Function Governance.read_config_bool_slot(string) (contracts/mosaic-alpha-contracts/Governance/Governance.sol#346) is not in mixedCase
Parameter Governance.read_config_bool_slot(string)._name (contracts/mosaic-alpha-contracts/Governance/Governance.sol#346) is not in mixedCase
Function Governance.read_config_address_slot(string) (contracts/mosaic-alpha-contracts/Governance/Governance.sol#347) is not in mixedCase
Parameter Governance.read_config_address_slot(string)._name (contracts/mosaic-alpha-contracts/Governance/Governance.sol#347) is not in mixedCase
Function Governance.read_config_uint256_slot(string) (contracts/mosaic-alpha-contracts/Governance/Governance.sol#348) is not in mixedCase
Parameter Governance.read_config_uint256_slot(string)._name (contracts/mosaic-alpha-contracts/Governance/Governance.sol#348) is not in mixedCase
Function Governance.read_config_bytes32_slot(string) (contracts/mosaic-alpha-contracts/Governance/Governance.sol#349) is not in mixedCase
Parameter Governance.read_config_bytes32_slot(string)._name (contracts/mosaic-alpha-contracts/Governance/Governance.sol#349) is not in mixedCase
Function Governance.read_config_Managers_batched(string,uint256[]) (contracts/mosaic-alpha-contracts/Governance/Governance.sol#351-357) is not in mixedCase
Parameter Governance.read_config_Managers_batched(string,uint256[])._name (contracts/mosaic-alpha-contracts/Governance/Governance.sol#351) is not in mixedCase
Parameter Governance.read_config_Managers_batched(string,uint256[])._ids (contracts/mosaic-alpha-contracts/Governance/Governance.sol#351) is not in mixedCase
Function Governance.read_config_bool_slot_batched(string,uint256[]) (contracts/mosaic-alpha-contracts/Governance/Governance.sol#359-365) is not in mixedCase
Parameter Governance.read_config_bool_slot_batched(string,uint256[])._name (contracts/mosaic-alpha-contracts/Governance/Governance.sol#359) is not in mixedCase
Parameter Governance.read_config_bool_slot_batched(string,uint256[])._ids (contracts/mosaic-alpha-contracts/Governance/Governance.sol#359) is not in mixedCase
Function Governance.read_config_address_slot_batched(string,uint256[]) (contracts/mosaic-alpha-contracts/Governance/Governance.sol#367-373) is not in mixedCase
Parameter Governance.read_config_address_slot_batched(string,uint256[])._name (contracts/mosaic-alpha-contracts/Governance/Governance.sol#367) is not in mixedCase
Parameter Governance.read_config_address_slot_batched(string,uint256[])._ids (contracts/mosaic-alpha-contracts/Governance/Governance.sol#367) is not in mixedCase
Function Governance.read_config_uint256_slot_batched(string,uint256[]) (contracts/mosaic-alpha-contracts/Governance/Governance.sol#375-381) is not in mixedCase
Parameter Governance.read_config_uint256_slot_batched(string,uint256[])._name (contracts/mosaic-alpha-contracts/Governance/Governance.sol#375) is not in mixedCase
Parameter Governance.read_config_uint256_slot_batched(string,uint256[])._ids (contracts/mosaic-alpha-contracts/Governance/Governance.sol#375) is not in mixedCase
Function Governance.read_config_bytes32_slot_batched(string,uint256[]) (contracts/mosaic-alpha-contracts/Governance/Governance.sol#383-389) is not in mixedCase
Parameter Governance.read_config_bytes32_slot_batched(string,uint256[])._name (contracts/mosaic-alpha-contracts/Governance/Governance.sol#383) is not in mixedCase
Parameter Governance.read_config_bytes32_slot_batched(string,uint256[])._ids (contracts/mosaic-alpha-contracts/Governance/Governance.sol#383) is not in mixedCase
Function Governance.read_invoteConfig_core(string) (contracts/mosaic-alpha-contracts/Governance/Governance.sol#394-403) is not in mixedCase
Parameter Governance.read_invoteConfig_core(string)._name (contracts/mosaic-alpha-contracts/Governance/Governance.sol#394) is not in mixedCase
Function Governance.read_invoteConfig_name(string) (contracts/mosaic-alpha-contracts/Governance/Governance.sol#404) is not in mixedCase
Parameter Governance.read_invoteConfig_name(string)._name (contracts/mosaic-alpha-contracts/Governance/Governance.sol#404) is not in mixedCase
Function Governance.read_invoteConfig_emergencyStatus(string) (contracts/mosaic-alpha-contracts/Governance/Governance.sol#405) is not in mixedCase
Parameter Governance.read_invoteConfig_emergencyStatus(string)._name (contracts/mosaic-alpha-contracts/Governance/Governance.sol#405) is not in mixedCase
Function Governance.read_invoteConfig_governAddress(string) (contracts/mosaic-alpha-contracts/Governance/Governance.sol#406) is not in mixedCase
Parameter Governance.read_invoteConfig_governAddress(string)._name (contracts/mosaic-alpha-contracts/Governance/Governance.sol#406) is not in mixedCase
Function Governance.read_invoteConfig_Managers(string) (contracts/mosaic-alpha-contracts/Governance/Governance.sol#407) is not in mixedCase
Parameter Governance.read_invoteConfig_Managers(string)._name (contracts/mosaic-alpha-contracts/Governance/Governance.sol#407) is not in mixedCase
Function Governance.read_invoteConfig_boolslot(string) (contracts/mosaic-alpha-contracts/Governance/Governance.sol#408) is not in mixedCase
Parameter Governance.read_invoteConfig_boolslot(string)._name (contracts/mosaic-alpha-contracts/Governance/Governance.sol#408) is not in mixedCase
Function Governance.read_invoteConfig_address_slot(string) (contracts/mosaic-alpha-contracts/Governance/Governance.sol#409) is not in mixedCase
Parameter Governance.read_invoteConfig_address_slot(string)._name (contracts/mosaic-alpha-contracts/Governance/Governance.sol#409) is not in mixedCase
Function Governance.read_invoteConfig_uint256_slot(string) (contracts/mosaic-alpha-contracts/Governance/Governance.sol#410) is not in mixedCase
Parameter Governance.read_invoteConfig_uint256_slot(string)._name (contracts/mosaic-alpha-contracts/Governance/Governance.sol#410) is not in mixedCase
Function Governance.read_invoteConfig_bytes32_slot(string) (contracts/mosaic-alpha-contracts/Governance/Governance.sol#411) is not in mixedCase
Parameter Governance.read_invoteConfig_bytes32_slot(string)._name (contracts/mosaic-alpha-contracts/Governance/Governance.sol#411) is not in mixedCase
Function Governance.propose_action(uint256,address,bytes) (contracts/mosaic-alpha-contracts/Governance/Governance.sol#417-430) is not in mixedCase
Parameter Governance.propose_action(uint256,address,bytes)._id (contracts/mosaic-alpha-contracts/Governance/Governance.sol#417) is not in mixedCase
Parameter Governance.propose_action(uint256,address,bytes)._trigger_address (contracts/mosaic-alpha-contracts/Governance/Governance.sol#417) is not in mixedCase
Parameter Governance.propose_action(uint256,address,bytes)._calldata (contracts/mosaic-alpha-contracts/Governance/Governance.sol#417) is not in mixedCase
Function Governance.support_actions(uint256) (contracts/mosaic-alpha-contracts/Governance/Governance.sol#433-441) is not in mixedCase
Parameter Governance.support_actions(uint256)._id (contracts/mosaic-alpha-contracts/Governance/Governance.sol#433) is not in mixedCase
Function Governance.trigger_action(uint256) (contracts/mosaic-alpha-contracts/Governance/Governance.sol#445-454) is not in mixedCase
Parameter Governance.trigger_action(uint256)._id (contracts/mosaic-alpha-contracts/Governance/Governance.sol#445) is not in mixedCase
Parameter Governance.generator(string)._func (contracts/mosaic-alpha-contracts/Governance/Governance.sol#457) is not in mixedCase
Function Governance.update_All(address[]) (contracts/mosaic-alpha-contracts/Governance/Governance.sol#465-468) is not in mixedCase
Parameter Governance.update_All(address[])._addresses (contracts/mosaic-alpha-contracts/Governance/Governance.sol#465) is not in mixedCase
Function Governance.selfManageMe_All() (contracts/mosaic-alpha-contracts/Governance/Governance.sol#471-475) is not in mixedCase
Function Governance.execute_Manage(address) (contracts/mosaic-alpha-contracts/Governance/Governance.sol#478-480) is not in mixedCase
Parameter Governance.execute_Manage(address)._contractA (contracts/mosaic-alpha-contracts/Governance/Governance.sol#478) is not in mixedCase
Function Governance._execute_Manage(address) (contracts/mosaic-alpha-contracts/Governance/Governance.sol#482-485) is not in mixedCase
Parameter Governance._execute_Manage(address)._contractA (contracts/mosaic-alpha-contracts/Governance/Governance.sol#482) is not in mixedCase
Function Governance.execute_batch_Manage(address[]) (contracts/mosaic-alpha-contracts/Governance/Governance.sol#488-492) is not in mixedCase
Parameter Governance.execute_batch_Manage(address[])._contracts (contracts/mosaic-alpha-contracts/Governance/Governance.sol#488) is not in mixedCase
Function Governance.execute_ManageBytes(address,string,bytes) (contracts/mosaic-alpha-contracts/Governance/Governance.sol#495-497) is not in mixedCase
Parameter Governance.execute_ManageBytes(address,string,bytes)._contractA (contracts/mosaic-alpha-contracts/Governance/Governance.sol#495) is not in mixedCase
Parameter Governance.execute_ManageBytes(address,string,bytes)._call (contracts/mosaic-alpha-contracts/Governance/Governance.sol#495) is not in mixedCase
Parameter Governance.execute_ManageBytes(address,string,bytes)._data (contracts/mosaic-alpha-contracts/Governance/Governance.sol#495) is not in mixedCase
Function Governance.execute_batch_ManageBytes(address[],string[],bytes[]) (contracts/mosaic-alpha-contracts/Governance/Governance.sol#499-505) is not in mixedCase
Parameter Governance.execute_batch_ManageBytes(address[],string[],bytes[])._contracts (contracts/mosaic-alpha-contracts/Governance/Governance.sol#499) is not in mixedCase
Parameter Governance.execute_batch_ManageBytes(address[],string[],bytes[])._calls (contracts/mosaic-alpha-contracts/Governance/Governance.sol#499) is not in mixedCase
Parameter Governance.execute_batch_ManageBytes(address[],string[],bytes[])._datas (contracts/mosaic-alpha-contracts/Governance/Governance.sol#499) is not in mixedCase
Function Governance._execute_ManageBytes(address,string,bytes) (contracts/mosaic-alpha-contracts/Governance/Governance.sol#507-524) is not in mixedCase

```
Parameter Governance._execute_ManageBytes(address,string,bytes)._contractA (contracts/mosaic-alpha-contracts/Governance/Governance.sol#507) is not in
mixedCase
Parameter Governance._execute_ManageBytes(address,string,bytes)._call (contracts/mosaic-alpha-contracts/Governance/Governance.sol#507) is not in mixedCase
Parameter Governance._execute_ManageBytes(address,string,bytes)._data (contracts/mosaic-alpha-contracts/Governance/Governance.sol#507) is not in mixedCase
Function Governance.execute_ManageList(address,string,address[]) (contracts/mosaic-alpha-contracts/Governance/Governance.sol#527-531) is not in mixedCase
Parameter Governance.execute_ManageList(address,string,address[])._contractA (contracts/mosaic-alpha-contracts/Governance/Governance.sol#527) is not in
mixedCase
Parameter Governance.execute_ManageList(address,string,address[])._funcName (contracts/mosaic-alpha-contracts/Governance/Governance.sol#527) is not in
mixedCase
Parameter Governance.execute_ManageList(address,string,address[]).address_array (contracts/mosaic-alpha-contracts/Governance/Governance.sol#527) is not in
mixedCase
Function Governance.execute_Vault_update(address) (contracts/mosaic-alpha-contracts/Governance/Governance.sol#534-536) is not in mixedCase
Parameter Governance.execute_Vault_update(address)._vaultAddress (contracts/mosaic-alpha-contracts/Governance/Governance.sol#534) is not in mixedCase
Variable Governance.curator_proportions (contracts/mosaic-alpha-contracts/Governance/Governance.sol#15) is not in mixedCase
Variable Governance.Configuration (contracts/mosaic-alpha-contracts/Governance/Governance.sol#19) is not in mixedCase
Variable Governance.ID_to_name (contracts/mosaic-alpha-contracts/Governance/Governance.sol#21) is not in mixedCase
Variable Governance.conf_curator_timer (contracts/mosaic-alpha-contracts/Governance/Governance.sol#23) is not in mixedCase
Variable Governance.conf_votes (contracts/mosaic-alpha-contracts/Governance/Governance.sol#24) is not in mixedCase
Variable Governance.conf_time_limit (contracts/mosaic-alpha-contracts/Governance/Governance.sol#25) is not in mixedCase
Variable Governance.conf_counter (contracts/mosaic-alpha-contracts/Governance/Governance.sol#26) is not in mixedCase
Constant Governance.Core (contracts/mosaic-alpha-contracts/Governance/Governance.sol#40) is not in UPPER_CASE_WITH_UNDERSCORES
Variable Governance.action_curator_timer (contracts/mosaic-alpha-contracts/Governance/Governance.sol#43) is not in mixedCase
Variable Governance.action_id_to_vote (contracts/mosaic-alpha-contracts/Governance/Governance.sol#44) is not in mixedCase
Variable Governance.action_time_limit (contracts/mosaic-alpha-contracts/Governance/Governance.sol#45) is not in mixedCase
Variable Governance.action_can_be_triggered_by (contracts/mosaic-alpha-contracts/Governance/Governance.sol#46) is not in mixedCase
Variable Governance.action_id_to_calldata (contracts/mosaic-alpha-contracts/Governance/Governance.sol#50) is not in mixedCase
Variable Governance.action_threshold (contracts/mosaic-alpha-contracts/Governance/Governance.sol#53) is not in mixedCase
Variable Governance.vote_time_threshold (contracts/mosaic-alpha-contracts/Governance/Governance.sol#54) is not in mixedCase
Variable Governance.vote_conf_time_threshold (contracts/mosaic-alpha-contracts/Governance/Governance.sol#55) is not in mixedCase
Function Governed.LiveFun() (contracts/mosaic-alpha-contracts/Governance/Governed.sol#45-47) is not in mixedCase
Modifier Governed.Live() (contracts/mosaic-alpha-contracts/Governance/Governed.sol#33-36) is not in mixedCase
Function IGovernance.propose_action(uint256,address,bytes) (contracts/mosaic-alpha-contracts/Interfaces/IGovernance.sol#5) is not in mixedCase
Parameter IGovernance.propose_action(uint256,address,bytes)._trigger_address (contracts/mosaic-alpha-contracts/Interfaces/IGovernance.sol#5) is not in
mixedCase
Function IGovernance.support_actions(uint256) (contracts/mosaic-alpha-contracts/Interfaces/IGovernance.sol#6) is not in mixedCase
Function IGovernance.trigger_action(uint256) (contracts/mosaic-alpha-contracts/Interfaces/IGovernance.sol#7) is not in mixedCase
Function IGovernance.transfer_proportion(address,uint256) (contracts/mosaic-alpha-contracts/Interfaces/IGovernance.sol#8) is not in mixedCase
Function IGovernance.read_core_Running() (contracts/mosaic-alpha-contracts/Interfaces/IGovernance.sol#10) is not in mixedCase
Function IGovernance.read_core_govAddr() (contracts/mosaic-alpha-contracts/Interfaces/IGovernance.sol#11) is not in mixedCase
Function IGovernance.read_core_managers() (contracts/mosaic-alpha-contracts/Interfaces/IGovernance.sol#12) is not in mixedCase
Function IGovernance.read_core_owners() (contracts/mosaic-alpha-contracts/Interfaces/IGovernance.sol#13) is not in mixedCase
Function IGovernance.read_config_core(string) (contracts/mosaic-alpha-contracts/Interfaces/IGovernance.sol#15) is not in mixedCase
Function IGovernance.read_config_emergencyStatus(string) (contracts/mosaic-alpha-contracts/Interfaces/IGovernance.sol#16) is not in mixedCase
Function IGovernance.read_config_governAddress(string) (contracts/mosaic-alpha-contracts/Interfaces/IGovernance.sol#17) is not in mixedCase
Function IGovernance.read_config_Managers(string) (contracts/mosaic-alpha-contracts/Interfaces/IGovernance.sol#18) is not in mixedCase
Function IGovernance.read_config_bool_slot(string) (contracts/mosaic-alpha-contracts/Interfaces/IGovernance.sol#20) is not in mixedCase
Function IGovernance.read_config_address_slot(string) (contracts/mosaic-alpha-contracts/Interfaces/IGovernance.sol#21) is not in mixedCase
Function IGovernance.read_config_uint256_slot(string) (contracts/mosaic-alpha-contracts/Interfaces/IGovernance.sol#22) is not in mixedCase
Function IGovernance.read_config_bytes32_slot(string) (contracts/mosaic-alpha-contracts/Interfaces/IGovernance.sol#23) is not in mixedCase
Function IGovernance.read_invoteConfig_core(string) (contracts/mosaic-alpha-contracts/Interfaces/IGovernance.sol#25) is not in mixedCase
Function IGovernance.read_invoteConfig_name(string) (contracts/mosaic-alpha-contracts/Interfaces/IGovernance.sol#26) is not in mixedCase
Function IGovernance.read_invoteConfig_emergencyStatus(string) (contracts/mosaic-alpha-contracts/Interfaces/IGovernance.sol#27) is not in mixedCase
Function IGovernance.read_invoteConfig_governAddress(string) (contracts/mosaic-alpha-contracts/Interfaces/IGovernance.sol#28) is not in mixedCase
Function IGovernance.read_invoteConfig_Managers(string) (contracts/mosaic-alpha-contracts/Interfaces/IGovernance.sol#29) is not in mixedCase
Function IGovernance.read_invoteConfig_boolslot(string) (contracts/mosaic-alpha-contracts/Interfaces/IGovernance.sol#30) is not in mixedCase
Function IGovernance.read_invoteConfig_address_slot(string) (contracts/mosaic-alpha-contracts/Interfaces/IGovernance.sol#31) is not in mixedCase
Function IGovernance.read_invoteConfig_uint256_slot(string) (contracts/mosaic-alpha-contracts/Interfaces/IGovernance.sol#32) is not in mixedCase
Function IGovernance.read_invoteConfig_bytes32_slot(string) (contracts/mosaic-alpha-contracts/Interfaces/IGovernance.sol#33) is not in mixedCase
Function IGovernance.propose_config(string,bool,address,address[],uint256,bytes32) (contracts/mosaic-alpha-contracts/Interfaces/IGovernance.sol#35) is not
in mixedCase
Parameter IGovernance.propose_config(string,bool,address,address[],uint256,bytes32)._bool_val (contracts/mosaic-alpha-
contracts/Interfaces/IGovernance.sol#35) is not in mixedCase
Parameter IGovernance.propose_config(string,bool,address,address[],uint256,bytes32)._address_val (contracts/mosaic-alpha-
contracts/Interfaces/IGovernance.sol#35) is not in mixedCase
Parameter IGovernance.propose_config(string,bool,address,address[],uint256,bytes32)._address_list (contracts/mosaic-alpha-
contracts/Interfaces/IGovernance.sol#35) is not in mixedCase
Parameter IGovernance.propose_config(string,bool,address,address[],uint256,bytes32)._uint256_val (contracts/mosaic-alpha-
contracts/Interfaces/IGovernance.sol#35) is not in mixedCase
Parameter IGovernance.propose_config(string,bool,address,address[],uint256,bytes32)._bytes32_val (contracts/mosaic-alpha-
contracts/Interfaces/IGovernance.sol#35) is not in mixedCase
Function IGovernance.support_config_proposal(uint256,string) (contracts/mosaic-alpha-contracts/Interfaces/IGovernance.sol#36) is not in mixedCase
Event ISwapstokenAdded(address) (contracts/mosaic-alpha-contracts/Interfaces/ISwaps.sol#44) is not in CapWords
Event ISwapstokenRemoved(address) (contracts/mosaic-alpha-contracts/Interfaces/ISwaps.sol#45) is not in CapWords
Parameter ISwaps.quickQuoteMint(address,address,uint256).LPTokensRequested (contracts/mosaic-alpha-contracts/Interfaces/ISwaps.sol#10) is not in mixedCase
Parameter ISwaps.quickQuoteBurn(address,address,uint256).LPTokensToBurn (contracts/mosaic-alpha-contracts/Interfaces/ISwaps.sol#12) is not in mixedCase
Parameter ISwaps.quoteMint(address,address[][],uint256).LPTokensRequested (contracts/mosaic-alpha-contracts/Interfaces/ISwaps.sol#18) is not in mixedCase
Parameter ISwaps.quoteBurn(address,address[][],uint256).LPTokensToBurn (contracts/mosaic-alpha-contracts/Interfaces/ISwaps.sol#20) is not in mixedCase
Parameter ISwaps.mint(address,address,address[][],uint256,uint256,address,uint256).LPTokensRequested (contracts/mosaic-alpha-
contracts/Interfaces/ISwaps.sol#23) is not in mixedCase
Parameter ISwaps.burn(address,address,address[][],uint256,uint256,uint256).LPTokens (contracts/mosaic-alpha-contracts/Interfaces/ISwaps.sol#26) is not in
mixedCase
Parameter ISwaps.quickMint(address,address,uint256,uint256,address,uint256).LPTokensRequested (contracts/mosaic-alpha-contracts/Interfaces/ISwaps.sol#29)
is not in mixedCase
Parameter ISwaps.quickBurn(address,address,uint256,uint256,uint256).LPTokens (contracts/mosaic-alpha-contracts/Interfaces/ISwaps.sol#32) is not in
mixedCase
Parameter Vault.getAuction(uint256)._id (contracts/mosaic-alpha-contracts/Vault/Vault.sol#61) is not in mixedCase
Parameter Vault.getPoolState(address)._pool (contracts/mosaic-alpha-contracts/Vault/Vault.sol#67) is not in mixedCase
Function Vault.LiveFun(address) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#141-143) is not in mixedCase
Parameter Vault.LiveFun(address)._a (contracts/mosaic-alpha-contracts/Vault/Vault.sol#141) is not in mixedCase
Parameter Vault.isAdmin(address)._address (contracts/mosaic-alpha-contracts/Vault/Vault.sol#233) is not in mixedCase
Function Vault.AddRemoveAdmin(address,bool) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#248-253) is not in mixedCase
Parameter Vault.AddRemoveAdmin(address,bool)._address (contracts/mosaic-alpha-contracts/Vault/Vault.sol#248) is not in mixedCase
Parameter Vault.AddRemoveAdmin(address,bool)._ShouldBeAdmin (contracts/mosaic-alpha-contracts/Vault/Vault.sol#248) is not in mixedCase
Function Vault.AddRemoveBoostedPool(address,bool) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#258-264) is not in mixedCase
Parameter Vault.AddRemoveBoostedPool(address,bool)._address (contracts/mosaic-alpha-contracts/Vault/Vault.sol#258) is not in mixedCase
Parameter Vault.AddRemoveBoostedPool(address,bool)._ShouldBeBoosted (contracts/mosaic-alpha-contracts/Vault/Vault.sol#258) is not in mixedCase
Function Vault.AddRemoveZeroFeeAddress(address,bool) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#266-270) is not in mixedCase
Parameter Vault.AddRemoveZeroFeeAddress(address,bool)._address (contracts/mosaic-alpha-contracts/Vault/Vault.sol#266) is not in mixedCase
Parameter Vault.AddRemoveZeroFeeAddress(address,bool)._ShouldBeZeroFee (contracts/mosaic-alpha-contracts/Vault/Vault.sol#266) is not in mixedCase
Parameter Vault.setWhitelistedTokens(address[],bool[])._tokens (contracts/mosaic-alpha-contracts/Vault/Vault.sol#353) is not in mixedCase
Parameter Vault.setWhitelistedTokens(address[],bool[])._whitelisted (contracts/mosaic-alpha-contracts/Vault/Vault.sol#353) is not in mixedCase
Parameter Vault.isTokenWhitelisted(address)._token (contracts/mosaic-alpha-contracts/Vault/Vault.sol#396) is not in mixedCase
Parameter Vault.getInternalBalance(address,address)._user (contracts/mosaic-alpha-contracts/Vault/Vault.sol#419) is not in mixedCase
Parameter Vault.getInternalBalance(address,address)._token (contracts/mosaic-alpha-contracts/Vault/Vault.sol#419) is not in mixedCase
Parameter Vault.allocateTrailingAndPerformanceFee(address)._pool (contracts/mosaic-alpha-contracts/Vault/Vault.sol#576) is not in mixedCase
Parameter Vault.registerPool(address,address,address)._pool (contracts/mosaic-alpha-contracts/Vault/Vault.sol#721) is not in mixedCase
Parameter Vault.registerPool(address,address,address)._user (contracts/mosaic-alpha-contracts/Vault/Vault.sol#721) is not in mixedCase
Parameter Vault.registerPool(address,address,address)._referredBy (contracts/mosaic-alpha-contracts/Vault/Vault.sol#721) is not in mixedCase
Parameter Vault.registerTokens(address[],bool)._newTokens (contracts/mosaic-alpha-contracts/Vault/Vault.sol#743) is not in mixedCase
Parameter Vault.addInitialLiquidity(uint32,address[],uint256[],address,bool)._poolId (contracts/mosaic-alpha-contracts/Vault/Vault.sol#779) is not in
mixedCase
Parameter Vault.addInitialLiquidity(uint32,address[],uint256[],address,bool)._tokens (contracts/mosaic-alpha-contracts/Vault/Vault.sol#779) is not in
mixedCase
Parameter Vault.addInitialLiquidity(uint32,address[],uint256[],address,bool)._liquidity (contracts/mosaic-alpha-contracts/Vault/Vault.sol#779) is not in
mixedCase
Parameter Vault.addInitialLiquidity(uint32,address[],uint256[],address,bool)._lpTo (contracts/mosaic-alpha-contracts/Vault/Vault.sol#779) is not in
mixedCase
```

```
Parameter Vault.deregisterToken(address,uint256)._tokenAddress (contracts/mosaic-alpha-contracts/Vault/Vault.sol#818) is not in mixedCase
Parameter Vault.deregisterToken(address,uint256)._remainingAmount (contracts/mosaic-alpha-contracts/Vault/Vault.sol#818) is not in mixedCase
Function Vault.Mint(uint32,uint256,uint256[],address,address,bool,uint256,uint256) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#841-904) is not in
mixedCase
Parameter Vault.Mint(uint32,uint256,uint256[],address,address,bool,uint256,uint256).LPTokensRequested (contracts/mosaic-alpha-
contracts/Vault/Vault.sol#841) is not in mixedCase
Function Vault.Burn(uint32,uint256,uint256[],bool,uint256,address) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#913-951) is not in mixedCase
Parameter Vault.Burn(uint32,uint256,uint256[],bool,uint256,address).LPTokensToBurn (contracts/mosaic-alpha-contracts/Vault/Vault.sol#913) is not in
mixedCase
Parameter Vault.getBidPriceAt(uint256,uint32)._now (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1122) is not in mixedCase
Parameter Vault.emergencyWithdraw(address,address,bool)._user (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1330) is not in mixedCase
Parameter Vault.emergencyWithdrawAll(address,address,uint256)._token (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1364) is not in mixedCase
Parameter Vault.emergencyWithdrawAll(address,address,uint256)._to (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1364) is not in mixedCase
Parameter Vault.emergencyWithdrawAll(address,address,uint256)._amount (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1364) is not in mixedCase
Parameter Vault.emergencyWithdrawPool(address,address,address,uint256)._pool (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1372) is not in mixedCase
Parameter Vault.emergencyWithdrawPool(address,address,address,uint256)._token (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1372) is not in mixedCase
Parameter Vault.emergencyWithdrawPool(address,address,address,uint256)._to (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1372) is not in mixedCase
Parameter Vault.emergencyWithdrawPool(address,address,address,uint256)._amount (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1372) is not in mixedCase
Variable Vault._poolIdToAddress (contracts/mosaic-alpha-contracts/Vault/Vault.sol#57) is not in mixedCase
Variable Vault._poolAddressToId (contracts/mosaic-alpha-contracts/Vault/Vault.sol#58) is not in mixedCase
Variable Vault._internalBalance (contracts/mosaic-alpha-contracts/Vault/Vault.sol#416) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Parameter Fees.getUserPoolFeeStatus(address,address)._user (contracts/mosaic-alpha-contracts/Fees/Fees.sol#69) is not in mixedCase
Parameter Fees.getUserPoolFeeStatus(address,address)._pool (contracts/mosaic-alpha-contracts/Fees/Fees.sol#69) is not in mixedCase
Parameter Fees.getPoolFeeStatus(address)._pool (contracts/mosaic-alpha-contracts/Fees/Fees.sol#73) is not in mixedCase
Parameter Fees.claimUserClaimableFee(address,address)._user (contracts/mosaic-alpha-contracts/Fees/Fees.sol#205) is not in mixedCase
Parameter Fees.claimUserClaimableFee(address,address)._pool (contracts/mosaic-alpha-contracts/Fees/Fees.sol#205) is not in mixedCase
Parameter Fees.claimTraderClaimableFee(address)._pool (contracts/mosaic-alpha-contracts/Fees/Fees.sol#308) is not in mixedCase
Parameter Fees.allocateBuyFee(address,address,address,uint256)._pool (contracts/mosaic-alpha-contracts/Fees/Fees.sol#417) is not in mixedCase
Parameter Fees.allocateBuyFee(address,address,address,uint256)._buyer (contracts/mosaic-alpha-contracts/Fees/Fees.sol#417) is not in mixedCase
Parameter Fees.allocateBuyFee(address,address,address,uint256)._trader (contracts/mosaic-alpha-contracts/Fees/Fees.sol#417) is not in mixedCase
Parameter Fees.allocateBuyFee(address,address,address,uint256)._buyFeeLpTokens (contracts/mosaic-alpha-contracts/Fees/Fees.sol#417) is not in mixedCase
Parameter Fees.allocateTrailingFee(address,address,uint256,uint256,address)._pool (contracts/mosaic-alpha-contracts/Fees/Fees.sol#455) is not in mixedCase
Parameter Fees.allocateTrailingFee(address,address,uint256,uint256,address)._trader (contracts/mosaic-alpha-contracts/Fees/Fees.sol#455) is not in
mixedCase
Parameter Fees.allocateTrailingFee(address,address,uint256,uint256,address)._feeAmount (contracts/mosaic-alpha-contracts/Fees/Fees.sol#455) is not in
mixedCase
Parameter Fees.allocateTrailingFee(address,address,uint256,uint256,address)._totalSupplyBefore (contracts/mosaic-alpha-contracts/Fees/Fees.sol#455) is not
in mixedCase
Parameter Fees.allocateTrailingFee(address,address,uint256,uint256,address)._executor (contracts/mosaic-alpha-contracts/Fees/Fees.sol#455) is not in
mixedCase
Parameter Fees.allocatePerformanceFee(address,address,uint256,uint256,address)._pool (contracts/mosaic-alpha-contracts/Fees/Fees.sol#499) is not in
mixedCase
Parameter Fees.allocatePerformanceFee(address,address,uint256,uint256,address)._trader (contracts/mosaic-alpha-contracts/Fees/Fees.sol#499) is not in
mixedCase
Parameter Fees.allocatePerformanceFee(address,address,uint256,uint256,address)._feeAmount (contracts/mosaic-alpha-contracts/Fees/Fees.sol#499) is not in
mixedCase
Parameter Fees.allocatePerformanceFee(address,address,uint256,uint256,address)._totalSupplyBefore (contracts/mosaic-alpha-contracts/Fees/Fees.sol#499) is
not in mixedCase
Parameter Fees.allocatePerformanceFee(address,address,uint256,uint256,address)._executor (contracts/mosaic-alpha-contracts/Fees/Fees.sol#499) is not in
mixedCase
Parameter Fees.onBeforeTransfer(address,address,address,uint256,uint256,uint256,uint256,address,IFees.OnBeforeTransferPayload)._pool (contracts/mosaic-
alpha-contracts/Fees/Fees.sol#537) is not in mixedCase
Parameter Fees.onBeforeTransfer(address,address,address,uint256,uint256,uint256,uint256,address,IFees.OnBeforeTransferPayload)._from (contracts/mosaic-
alpha-contracts/Fees/Fees.sol#537) is not in mixedCase
Parameter Fees.onBeforeTransfer(address,address,address,uint256,uint256,uint256,uint256,address,IFees.OnBeforeTransferPayload)._to (contracts/mosaic-alpha-
contracts/Fees/Fees.sol#537) is not in mixedCase
Parameter Fees.onBeforeTransfer(address,address,address,uint256,uint256,uint256,uint256,address,IFees.OnBeforeTransferPayload)._fromBalanceBefore
(contracts/mosaic-alpha-contracts/Fees/Fees.sol#537) is not in mixedCase
Parameter Fees.onBeforeTransfer(address,address,address,uint256,uint256,uint256,uint256,address,IFees.OnBeforeTransferPayload)._toBalanceBefore
(contracts/mosaic-alpha-contracts/Fees/Fees.sol#537) is not in mixedCase
Parameter Fees.onBeforeTransfer(address,address,address,uint256,uint256,uint256,uint256,address,IFees.OnBeforeTransferPayload)._amount (contracts/mosaic-
alpha-contracts/Fees/Fees.sol#537) is not in mixedCase
Parameter Fees.onBeforeTransfer(address,address,address,uint256,uint256,uint256,uint256,address,IFees.OnBeforeTransferPayload)._totalSupplyBefore
(contracts/mosaic-alpha-contracts/Fees/Fees.sol#537) is not in mixedCase
Parameter Fees.onBeforeTransfer(address,address,address,uint256,uint256,uint256,uint256,address,IFees.OnBeforeTransferPayload)._trader (contracts/mosaic-
alpha-contracts/Fees/Fees.sol#537) is not in mixedCase
Parameter Fees.updateDiscountCurve(uint256[],uint32[])._x (contracts/mosaic-alpha-contracts/Fees/Fees.sol#656) is not in mixedCase
Parameter Fees.updateDiscountCurve(uint256[],uint32[])._y (contracts/mosaic-alpha-contracts/Fees/Fees.sol#656) is not in mixedCase
Parameter Fees.updateTraderShareCurve(uint256[],uint32[])._x (contracts/mosaic-alpha-contracts/Fees/Fees.sol#682) is not in mixedCase
Parameter Fees.updateTraderShareCurve(uint256[],uint32[])._y (contracts/mosaic-alpha-contracts/Fees/Fees.sol#682) is not in mixedCase
Parameter Fees.setExecutorShares(uint8,uint8,uint8)._share (contracts/mosaic-alpha-contracts/Fees/Fees.sol#704) is not in mixedCase
Parameter Fees.setExecutorShares(uint8,uint8,uint8)._user (contracts/mosaic-alpha-contracts/Fees/Fees.sol#704) is not in mixedCase
Parameter Fees.setExecutorShares(uint8,uint8,uint8)._trader (contracts/mosaic-alpha-contracts/Fees/Fees.sol#704) is not in mixedCase
Parameter Fees.setTraderClaimer(address)._a (contracts/mosaic-alpha-contracts/Fees/Fees.sol#710) is not in mixedCase
Parameter Fees.updateFeeRanges(IFees.MosaicFeeRanges)._newRanges (contracts/mosaic-alpha-contracts/Fees/Fees.sol#715) is not in mixedCase
Parameter Fees.updateFeeDistribution(IFees.MosaicFeeDistribution)._newDistribution (contracts/mosaic-alpha-contracts/Fees/Fees.sol#732) is not in mixedCase
Parameter Fees.updatePlatformFeeShares(IFees.MosaicPlatformFeeShares)._newShares (contracts/mosaic-alpha-contracts/Fees/Fees.sol#744) is not in mixedCase
Function Fees._calculateEffectiveBuyFeeDiscountAmount(uint256,uint16,uint32) (contracts/mosaic-alpha-contracts/Fees/Fees.sol#843-845) is not in mixedCase
Function Fees._calculateEffectiveTraderRevenueShareAmount(uint256,uint16,uint16,uint32) (contracts/mosaic-alpha-contracts/Fees/Fees.sol#856-858) is not in
mixedCase
Parameter Fees.userRankChanged(address,uint8)._user (contracts/mosaic-alpha-contracts/Fees/Fees.sol#865) is not in mixedCase
Parameter Fees.userRankChanged(address,uint8)._rank (contracts/mosaic-alpha-contracts/Fees/Fees.sol#865) is not in mixedCase
Parameter Fees.userStakeChanged(address,uint256)._user (contracts/mosaic-alpha-contracts/Fees/Fees.sol#874) is not in mixedCase
Parameter Fees.userStakeChanged(address,uint256)._amount (contracts/mosaic-alpha-contracts/Fees/Fees.sol#874) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Function ITradingBot.SetOperatorAddress(address) (contracts/mosaic-alpha-bot/Interfaces/Interfaces/ITradingBot.sol#18) is not in mixedCase
Function ITradingBot.SetVaultAddress(address) (contracts/mosaic-alpha-bot/Interfaces/Interfaces/ITradingBot.sol#19) is not in mixedCase
Function ITradingBot.SetPancakeAddress(address) (contracts/mosaic-alpha-bot/Interfaces/Interfaces/ITradingBot.sol#20) is not in mixedCase
Function ITradingBot.TradingStartFromPancake(ITradingBot.TradingInput) (contracts/mosaic-alpha-bot/Interfaces/Interfaces/ITradingBot.sol#22-24) is not in
mixedCase
Function ITradingBot.TradingStartFromMosaic(ITradingBot.TradingInput) (contracts/mosaic-alpha-bot/Interfaces/Interfaces/ITradingBot.sol#26-28) is not in
mixedCase
Function ITradingBot.GetAmountOutStartMosaic(ITradingBot.TradingInput,uint16) (contracts/mosaic-alpha-bot/Interfaces/Interfaces/ITradingBot.sol#30) is not
in mixedCase
Function ITradingBot.GetAmountOutStartPancake(ITradingBot.TradingInput,uint16) (contracts/mosaic-alpha-bot/Interfaces/Interfaces/ITradingBot.sol#31) is not
in mixedCase
Function ITradingBot.Transfer(address,uint256) (contracts/mosaic-alpha-bot/Interfaces/Interfaces/ITradingBot.sol#33) is not in mixedCase
Function TradingBot.SetOperatorAddress(address) (contracts/mosaic-alpha-bot/Interfaces/TradingBot.sol#26-29) is not in mixedCase
Function TradingBot.SetVaultAddress(address) (contracts/mosaic-alpha-bot/Interfaces/TradingBot.sol#31-34) is not in mixedCase
Function TradingBot.SetPancakeAddress(address) (contracts/mosaic-alpha-bot/Interfaces/TradingBot.sol#36-39) is not in mixedCase
Function TradingBot.GetAmountOutStartMosaic(ITradingBot.TradingInput,uint16) (contracts/mosaic-alpha-bot/Interfaces/TradingBot.sol#45-62) is not in
mixedCase
Function TradingBot.GetAmountOutStartPancake(ITradingBot.TradingInput,uint16) (contracts/mosaic-alpha-bot/Interfaces/TradingBot.sol#64-83) is not in
mixedCase
Function TradingBot.TradingStartFromPancake(ITradingBot.TradingInput) (contracts/mosaic-alpha-bot/Interfaces/TradingBot.sol#85-111) is not in mixedCase
Function TradingBot.TradingStartFromMosaic(ITradingBot.TradingInput) (contracts/mosaic-alpha-bot/Interfaces/TradingBot.sol#113-141) is not in mixedCase
Function TradingBot._Percentage(uint256,uint16) (contracts/mosaic-alpha-bot/Interfaces/TradingBot.sol#143-145) is not in mixedCase
Function TradingBot.Transfer(address,uint256) (contracts/mosaic-alpha-bot/Interfaces/TradingBot.sol#147-150) is not in mixedCase
Variable TradingBot._operatorAddress (contracts/mosaic-alpha-bot/Interfaces/TradingBot.sol#12) is not in mixedCase
Variable TradingBot._pancakeRouter (contracts/mosaic-alpha-bot/Interfaces/TradingBot.sol#14) is not in mixedCase
Variable TradingBot._vault (contracts/mosaic-alpha-bot/Interfaces/TradingBot.sol#15) is not in mixedCase
Parameter SimpleDeposit.changeRegister(address)._register (contracts/mosaic-alpha-contracts/Deposit/SimpleDeposit.sol#93) is not in mixedCase
Parameter SimpleDeposit.changeFees(address)._fees (contracts/mosaic-alpha-contracts/Deposit/SimpleDeposit.sol#101) is not in mixedCase
Parameter SimpleDeposit.changeAffiliate(address)._affiliate (contracts/mosaic-alpha-contracts/Deposit/SimpleDeposit.sol#109) is not in mixedCase
```

Parameter SimpleDeposit.setLockTime(uint64)._time (contracts/mosaic-alpha-contracts/Deposit/SimpleDeposit.sol#124) is not in mixedCase
Parameter SimpleDeposit.approveExternalAddress(address)._locker (contracts/mosaic-alpha-contracts/Deposit/SimpleDeposit.sol#137) is not in mixedCase
Parameter SimpleDeposit.disapproveExternalAddress(address)._locker (contracts/mosaic-alpha-contracts/Deposit/SimpleDeposit.sol#145) is not in mixedCase
Parameter SimpleDeposit.isApprovedExternalAddress(address,address)._account (contracts/mosaic-alpha-contracts/Deposit/SimpleDeposit.sol#152) is not in mixedCase
Parameter SimpleDeposit.isApprovedExternalAddress(address,address)._locker (contracts/mosaic-alpha-contracts/Deposit/SimpleDeposit.sol#152) is not in mixedCase
Parameter SimpleDeposit.deposit(uint256)._amount (contracts/mosaic-alpha-contracts/Deposit/SimpleDeposit.sol#167) is not in mixedCase
Parameter SimpleDeposit.depositTo(address,uint256)._recipient (contracts/mosaic-alpha-contracts/Deposit/SimpleDeposit.sol#171) is not in mixedCase
Parameter SimpleDeposit.depositTo(address,uint256)._amount (contracts/mosaic-alpha-contracts/Deposit/SimpleDeposit.sol#171) is not in mixedCase
Parameter SimpleDeposit.withdraw(uint256)._amount (contracts/mosaic-alpha-contracts/Deposit/SimpleDeposit.sol#181) is not in mixedCase
Parameter SimpleDeposit.userLock(uint256)._amount (contracts/mosaic-alpha-contracts/Deposit/SimpleDeposit.sol#186) is not in mixedCase
Parameter SimpleDeposit.userLock(uint256,address)._amount (contracts/mosaic-alpha-contracts/Deposit/SimpleDeposit.sol#191) is not in mixedCase
Parameter SimpleDeposit.userLock(uint256,address)._referredBy (contracts/mosaic-alpha-contracts/Deposit/SimpleDeposit.sol#191) is not in mixedCase
Parameter SimpleDeposit.userUnlock(uint256)._amount (contracts/mosaic-alpha-contracts/Deposit/SimpleDeposit.sol#196) is not in mixedCase
Parameter SimpleDeposit.depositAndLock(uint256)._amount (contracts/mosaic-alpha-contracts/Deposit/SimpleDeposit.sol#200) is not in mixedCase
Parameter SimpleDeposit.depositAndLock(uint256,address)._amount (contracts/mosaic-alpha-contracts/Deposit/SimpleDeposit.sol#205) is not in mixedCase
Parameter SimpleDeposit.depositAndLock(uint256,address)._referredBy (contracts/mosaic-alpha-contracts/Deposit/SimpleDeposit.sol#205) is not in mixedCase
Parameter SimpleDeposit.unlockAndWithdraw(uint256)._amount (contracts/mosaic-alpha-contracts/Deposit/SimpleDeposit.sol#210) is not in mixedCase
Parameter SimpleDeposit.externalLock(address,uint256)._account (contracts/mosaic-alpha-contracts/Deposit/SimpleDeposit.sol#217) is not in mixedCase
Parameter SimpleDeposit.externalLock(address,uint256)._amount (contracts/mosaic-alpha-contracts/Deposit/SimpleDeposit.sol#217) is not in mixedCase
Parameter SimpleDeposit.externalUnlock(address,uint256)._account (contracts/mosaic-alpha-contracts/Deposit/SimpleDeposit.sol#222) is not in mixedCase
Parameter SimpleDeposit.externalUnlock(address,uint256)._amount (contracts/mosaic-alpha-contracts/Deposit/SimpleDeposit.sol#222) is not in mixedCase
Parameter SimpleDeposit.getAccountBalance(address)._account (contracts/mosaic-alpha-contracts/Deposit/SimpleDeposit.sol#236) is not in mixedCase
Parameter SimpleDeposit.getAccountBalances(address)._account (contracts/mosaic-alpha-contracts/Deposit/SimpleDeposit.sol#241) is not in mixedCase
Parameter SimpleDeposit.getAccountLockedBalanceUnlockTimestamp(address)._account (contracts/mosaic-alpha-contracts/Deposit/SimpleDeposit.sol#246) is not in mixedCase
Parameter SimpleDeposit.getWithdrawableAccountLockedBalance(address)._account (contracts/mosaic-alpha-contracts/Deposit/SimpleDeposit.sol#250) is not in mixedCase
Parameter SimpleDeposit.getAccountLockedBalance(address)._account (contracts/mosaic-alpha-contracts/Deposit/SimpleDeposit.sol#256) is not in mixedCase
Parameter SimpleDeposit.getExternalLockedBalance(address)._account (contracts/mosaic-alpha-contracts/Deposit/SimpleDeposit.sol#262) is not in mixedCase
Variable SimpleDeposit._stakingToken (contracts/mosaic-alpha-contracts/Deposit/SimpleDeposit.sol#34) is not in mixedCase
Variable SimpleDeposit._affiliateContract (contracts/mosaic-alpha-contracts/Deposit/SimpleDeposit.sol#35) is not in mixedCase
Variable SimpleDeposit._feesContract (contracts/mosaic-alpha-contracts/Deposit/SimpleDeposit.sol#36) is not in mixedCase
Parameter FeeBurner.setFeeToken(address)._token (contracts/mosaic-alpha-contracts/Fees/FeeBurner.sol#78) is not in mixedCase
Parameter FeeBurner.setFeeTokenBurnable(bool)._burnable (contracts/mosaic-alpha-contracts/Fees/FeeBurner.sol#82) is not in mixedCase
Parameter FeeBurner.setFeeTo(address)._receiver (contracts/mosaic-alpha-contracts/Fees/FeeBurner.sol#86) is not in mixedCase
Function IVaultDelegated.DselfManageMe() (contracts/mosaic-alpha-contracts/Interfaces/IVaultDelegated.sol#5) is not in mixedCase
Variable Initializable._____gap (contracts/mosaic-alpha-contracts/Oracle/Initializable.sol#66) is not in mixedCase
Parameter SimpleMosaicOracle.initialize(address,address,address,address)._pancakeRouter (contracts/mosaic-alpha-contracts/Oracle/SimpleMosaicOracle.sol#31) is not in mixedCase
Parameter SimpleMosaicOracle.initialize(address,address,address,address)._stablecoin (contracts/mosaic-alpha-contracts/Oracle/SimpleMosaicOracle.sol#31) is not in mixedCase
Parameter SimpleMosaicOracle.initialize(address,address,address,address)._defaultMiddleHop (contracts/mosaic-alpha-contracts/Oracle/SimpleMosaicOracle.sol#31) is not in mixedCase
Parameter SimpleMosaicOracle.initialize(address,address,address,address)._vault (contracts/mosaic-alpha-contracts/Oracle/SimpleMosaicOracle.sol#31) is not in mixedCase
Parameter SimpleMosaicOracle.setDefaultMiddleHop(address)._defaultMiddleHop (contracts/mosaic-alpha-contracts/Oracle/SimpleMosaicOracle.sol#44) is not in mixedCase
Parameter SimpleMosaicOracle.getMiddleHop(address,address)._a (contracts/mosaic-alpha-contracts/Oracle/SimpleMosaicOracle.sol#80) is not in mixedCase
Parameter SimpleMosaicOracle.getMiddleHop(address,address)._b (contracts/mosaic-alpha-contracts/Oracle/SimpleMosaicOracle.sol#80) is not in mixedCase
Parameter SimpleMosaicOracle.getUsdPoolPricePerLp(address,IVault.TotalSupplyBase)._poolAddress (contracts/mosaic-alpha-contracts/Oracle/SimpleMosaicOracle.sol#178) is not in mixedCase
Parameter SimpleMosaicOracle.getUsdPoolPricePerLp(address,IVault.TotalSupplyBase)._supplyBase (contracts/mosaic-alpha-contracts/Oracle/SimpleMosaicOracle.sol#179) is not in mixedCase
Parameter SimpleMosaicOracle.consultUsdPoolPricePerLp(address,IVault.TotalSupplyBase)._poolAddress (contracts/mosaic-alpha-contracts/Oracle/SimpleMosaicOracle.sol#335) is not in mixedCase
Parameter SimpleMosaicOracle.consultUsdPoolPricePerLp(address,IVault.TotalSupplyBase)._supplyBase (contracts/mosaic-alpha-contracts/Oracle/SimpleMosaicOracle.sol#336) is not in mixedCase
Parameter PoolFactory.addSystemAddress(address)._a (contracts/mosaic-alpha-contracts/Pool/PoolFactory.sol#58) is not in mixedCase
Parameter PoolFactory.removeSystemAddress(address)._a (contracts/mosaic-alpha-contracts/Pool/PoolFactory.sol#61) is not in mixedCase
Parameter PoolFactory.setMinInitialLiquidityValue(uint256)._value (contracts/mosaic-alpha-contracts/Pool/PoolFactory.sol#269) is not in mixedCase
Parameter PoolFactory.setFeeBurner(address)._burner (contracts/mosaic-alpha-contracts/Pool/PoolFactory.sol#277) is not in mixedCase
Parameter PoolFactory.setCreatePoolFeeAmount(uint112)._a (contracts/mosaic-alpha-contracts/Pool/PoolFactory.sol#285) is not in mixedCase
Parameter QueryApi.setVaultaddress(address)._vaultAddr (contracts/mosaic-alpha-contracts/Query/QueryApi.sol#18) is not in mixedCase
Parameter Register.approvePools(address[])._poolAddrs (contracts/mosaic-alpha-contracts/Register/Register.sol#33) is not in mixedCase
Parameter Register.approvePoolsById(uint32[])._poolIds (contracts/mosaic-alpha-contracts/Register/Register.sol#39) is not in mixedCase
Parameter Register.revokePools(address[])._poolAddrs (contracts/mosaic-alpha-contracts/Register/Register.sol#45) is not in mixedCase
Parameter Register.revokePoolsById(uint32[])._poolIds (contracts/mosaic-alpha-contracts/Register/Register.sol#51) is not in mixedCase
Parameter Register.approveTraders(address[])._traders (contracts/mosaic-alpha-contracts/Register/Register.sol#69) is not in mixedCase
Parameter Register.revokeTraders(address[])._traders (contracts/mosaic-alpha-contracts/Register/Register.sol#75) is not in mixedCase
Parameter Register.isApprovedPool(uint32)._poolId (contracts/mosaic-alpha-contracts/Register/Register.sol#91) is not in mixedCase
Parameter Register.isApprovedPool(address)._poolAddr (contracts/mosaic-alpha-contracts/Register/Register.sol#95) is not in mixedCase
Parameter Register.isApprovedTrader(address)._address (contracts/mosaic-alpha-contracts/Register/Register.sol#99) is not in mixedCase
Parameter Register.isWhitelisted(address)._address (contracts/mosaic-alpha-contracts/Register/Register.sol#109) is not in mixedCase
Parameter Register.isBlacklisted(address)._address (contracts/mosaic-alpha-contracts/Register/Register.sol#118) is not in mixedCase
Parameter Register.addWhitelist(address)._address (contracts/mosaic-alpha-contracts/Register/Register.sol#130) is not in mixedCase
Parameter Register.addWhitelistBulk(address[])._address (contracts/mosaic-alpha-contracts/Register/Register.sol#142) is not in mixedCase
Parameter Register.addBlacklist(address)._address (contracts/mosaic-alpha-contracts/Register/Register.sol#156) is not in mixedCase
Parameter Register.removeWhitelist(address)._address (contracts/mosaic-alpha-contracts/Register/Register.sol#168) is not in mixedCase
Parameter Register.removeBlacklist(address)._address (contracts/mosaic-alpha-contracts/Register/Register.sol#178) is not in mixedCase
Parameter Affiliate.updateConfig(IAffiliate.AffiliateConfig)._newConfig (contracts/mosaic-alpha-contracts/User/Affiliate.sol#61) is not in mixedCase
Parameter Affiliate.getUserData(address)._a (contracts/mosaic-alpha-contracts/User/Affiliate.sol#69) is not in mixedCase
Parameter Affiliate.getLevelDetails(uint256)._idx (contracts/mosaic-alpha-contracts/User/Affiliate.sol#78) is not in mixedCase
Parameter Affiliate.setUserProfileContract(address)._profile (contracts/mosaic-alpha-contracts/User/Affiliate.sol#148) is not in mixedCase
Parameter Affiliate.setUserOracleContract(address)._oracle (contracts/mosaic-alpha-contracts/User/Affiliate.sol#152) is not in mixedCase
Parameter AffiliateBooster.setController(address)._a (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#226) is not in mixedCase
Parameter AffiliateBooster.setCallerAllowed(address,bool)._a (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#243) is not in mixedCase
Parameter AffiliateBooster.setCallerAllowed(address,bool)._allowed (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#243) is not in mixedCase
Parameter AffiliateBooster.setEpochWeights(uint8[])._weights (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#287) is not in mixedCase
Parameter AffiliateBooster.setFeeTo(address)._a (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#305) is not in mixedCase
Parameter AffiliateBooster.purchaseTicket(uint256)._c (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#313) is not in mixedCase
Parameter AffiliateBooster.purchaseTicket(address,uint256)._referredBy (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#322) is not in mixedCase
Parameter AffiliateBooster.purchaseTicket(address,uint256)._c (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#322) is not in mixedCase
Parameter AffiliateBooster.allocateTicket(address,address,uint256,uint256)._a (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#340) is not in mixedCase
Parameter AffiliateBooster.allocateTicket(address,address,uint256,uint256)._referredBy (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#340) is not in mixedCase
Parameter AffiliateBooster.allocateTicket(address,address,uint256,uint256)._c (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#340) is not in mixedCase
Parameter AffiliateBooster.allocateTicket(address,address,uint256,uint256)._price (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#340) is not in mixedCase
Contract Affiliate_Storage (contracts/mosaic-alpha-contracts/User/Affiliate_old.sol#10-136) is not in CapWords
Struct Affiliate_Storage.affiliate_struct (contracts/mosaic-alpha-contracts/User/Affiliate_old.sol#38-47) is not in CapWords
Event Affiliate_Storageadded_affiliate(string) (contracts/mosaic-alpha-contracts/User/Affiliate_old.sol#18) is not in CapWords
Event Affiliate_Storageaffiliate_data_updated(string) (contracts/mosaic-alpha-contracts/User/Affiliate_old.sol#19) is not in CapWords
Event Affiliate_Storageoracle_update(string) (contracts/mosaic-alpha-contracts/User/Affiliate_old.sol#20) is not in CapWords
Event Affiliate_Storagepaused_now(bool) (contracts/mosaic-alpha-contracts/User/Affiliate_old.sol#21) is not in CapWords
Function Affiliate_Storage.add_referred_user(address,address,address,uint256,uint256,uint256,uint256,uint256) (contracts/mosaic-alpha-contracts/User/Affiliate_old.sol#50-72) is not in mixedCase
Parameter Affiliate_Storage.add_referred_user(address,address,address,uint256,uint256,uint256,uint256,uint256)._referee (contracts/mosaic-alpha-contracts/User/Affiliate_old.sol#51) is not in mixedCase
Parameter Affiliate_Storage.add_referred_user(address,address,address,uint256,uint256,uint256,uint256,uint256)._referred_by (contracts/mosaic-alpha-contracts/User/Affiliate_old.sol#52) is not in mixedCase
Parameter Affiliate_Storage.add_referred_user(address,address,address,uint256,uint256,uint256,uint256,uint256)._referred_by_before (contracts/mosaic-alpha-contracts/User/Affiliate_old.sol#53) is not in mixedCase
Parameter Affiliate_Storage.add_referred_user(address,address,address,uint256,uint256,uint256,uint256,uint256)._referred_by_rank (contracts/mosaic-alpha-

contracts/User/Affiliate_old.sol#54) is not in mixedCase
Parameter Affiliate_Storage.add_referred_user(address,address,address,uint256,uint256,uint256,uint256,uint256)._referee_balance (contracts/mosaic-alpha-contracts/User/Affiliate_old.sol#55) is not in mixedCase
Parameter Affiliate_Storage.add_referred_user(address,address,address,uint256,uint256,uint256,uint256,uint256)._referee_count (contracts/mosaic-alpha-contracts/User/Affiliate_old.sol#56) is not in mixedCase
Parameter Affiliate_Storage.add_referred_user(address,address,address,uint256,uint256,uint256,uint256,uint256)._referee_stake_amount (contracts/mosaic-alpha-contracts/User/Affiliate_old.sol#57) is not in mixedCase
Parameter Affiliate_Storage.add_referred_user(address,address,address,uint256,uint256,uint256,uint256,uint256)._referred_by_balance (contracts/mosaic-alpha-contracts/User/Affiliate_old.sol#58) is not in mixedCase
Function Affiliate_Storage.update_ref_by_rank(address,uint256) (contracts/mosaic-alpha-contracts/User/Affiliate_old.sol#74-78) is not in mixedCase
Parameter Affiliate_Storage.update_ref_by_rank(address,uint256)._referee (contracts/mosaic-alpha-contracts/User/Affiliate_old.sol#74) is not in mixedCase
Parameter Affiliate_Storage.update_ref_by_rank(address,uint256)._referred_by_rank (contracts/mosaic-alpha-contracts/User/Affiliate_old.sol#74) is not in mixedCase
Function Affiliate_Storage.update_referee_balance(address,uint256) (contracts/mosaic-alpha-contracts/User/Affiliate_old.sol#80-84) is not in mixedCase
Parameter Affiliate_Storage.update_referee_balance(address,uint256)._referee (contracts/mosaic-alpha-contracts/User/Affiliate_old.sol#80) is not in mixedCase
Parameter Affiliate_Storage.update_referee_balance(address,uint256)._referee_balance (contracts/mosaic-alpha-contracts/User/Affiliate_old.sol#80) is not in mixedCase
Function Affiliate_Storage.update_referee_count(address,uint256) (contracts/mosaic-alpha-contracts/User/Affiliate_old.sol#86-90) is not in mixedCase
Parameter Affiliate_Storage.update_referee_count(address,uint256)._referee (contracts/mosaic-alpha-contracts/User/Affiliate_old.sol#86) is not in mixedCase
Parameter Affiliate_Storage.update_referee_count(address,uint256)._referee_count (contracts/mosaic-alpha-contracts/User/Affiliate_old.sol#86) is not in mixedCase
Function Affiliate_Storage.update_referee_stake(address,uint256) (contracts/mosaic-alpha-contracts/User/Affiliate_old.sol#92-96) is not in mixedCase
Parameter Affiliate_Storage.update_referee_stake(address,uint256)._referee (contracts/mosaic-alpha-contracts/User/Affiliate_old.sol#92) is not in mixedCase
Parameter Affiliate_Storage.update_referee_stake(address,uint256)._referee_stake_amount (contracts/mosaic-alpha-contracts/User/Affiliate_old.sol#92) is not in mixedCase
Function Affiliate_Storage.update_ref_by_balance(address,uint256) (contracts/mosaic-alpha-contracts/User/Affiliate_old.sol#98-102) is not in mixedCase
Parameter Affiliate_Storage.update_ref_by_balance(address,uint256)._referee (contracts/mosaic-alpha-contracts/User/Affiliate_old.sol#98) is not in mixedCase
Parameter Affiliate_Storage.update_ref_by_balance(address,uint256)._referred_by_balance (contracts/mosaic-alpha-contracts/User/Affiliate_old.sol#98) is not in mixedCase
Function Affiliate_Storage.update_referee_rank(address,uint256) (contracts/mosaic-alpha-contracts/User/Affiliate_old.sol#104-108) is not in mixedCase
Parameter Affiliate_Storage.update_referee_rank(address,uint256)._referee (contracts/mosaic-alpha-contracts/User/Affiliate_old.sol#104) is not in mixedCase
Parameter Affiliate_Storage.update_referee_rank(address,uint256)._rank (contracts/mosaic-alpha-contracts/User/Affiliate_old.sol#104) is not in mixedCase
Function Affiliate_Storage.change_oracle(address) (contracts/mosaic-alpha-contracts/User/Affiliate_old.sol#110-114) is not in mixedCase
Parameter Affiliate_Storage.change_oracle(address)._new_oracle (contracts/mosaic-alpha-contracts/User/Affiliate_old.sol#110) is not in mixedCase
Function Affiliate_Storage.pause_unpause(bool) (contracts/mosaic-alpha-contracts/User/Affiliate_old.sol#116-120) is not in mixedCase
Parameter Affiliate_Storage.pause_unpause(bool)._pause_state (contracts/mosaic-alpha-contracts/User/Affiliate_old.sol#116) is not in mixedCase
Function Affiliate_Storage.read_affiliate(address) (contracts/mosaic-alpha-contracts/User/Affiliate_old.sol#122-131) is not in mixedCase
Parameter Affiliate_Storage.read_affiliate(address)._address (contracts/mosaic-alpha-contracts/User/Affiliate_old.sol#122) is not in mixedCase
Variable Affiliate_Storage.user_count (contracts/mosaic-alpha-contracts/User/Affiliate_old.sol#16) is not in mixedCase
Modifier Affiliate_Storage.pause_state() (contracts/mosaic-alpha-contracts/User/Affiliate_old.sol#33-36) is not in mixedCase
Parameter SimpleAffiliateBoosterController.setTicketPrice(uint256)._price (contracts/mosaic-alpha-contracts/User/SimpleAffiliateBoosterController.sol#24) is not in mixedCase
Parameter SimpleAffiliateBoosterController.setFeeToken(address)._a (contracts/mosaic-alpha-contracts/User/SimpleAffiliateBoosterController.sol#28) is not in mixedCase
Parameter SimpleAffiliateBoosterController.setFeeTokenTo(address)._a (contracts/mosaic-alpha-contracts/User/SimpleAffiliateBoosterController.sol#32) is not in mixedCase
Parameter SimpleAffiliateBoosterController.setFeeTokenBurnable(bool)._b (contracts/mosaic-alpha-contracts/User/SimpleAffiliateBoosterController.sol#36) is not in mixedCase
Parameter SimpleAffiliateBoosterController.purchaseTicket(uint256)._c (contracts/mosaic-alpha-contracts/User/SimpleAffiliateBoosterController.sol#40) is not in mixedCase
Parameter SimpleAffiliateBoosterController.purchaseTicket(address,uint256)._referredBy (contracts/mosaic-alpha-contracts/User/SimpleAffiliateBoosterController.sol#44) is not in mixedCase
Parameter SimpleAffiliateBoosterController.purchaseTicket(address,uint256)._c (contracts/mosaic-alpha-contracts/User/SimpleAffiliateBoosterController.sol#44) is not in mixedCase
Parameter UserProfile.initializeRootNode(address)._user (contracts/mosaic-alpha-contracts/User/UserProfile.sol#90) is not in mixedCase
Parameter UserProfile.setAffiliateBooster(address)._a (contracts/mosaic-alpha-contracts/User/UserProfile.sol#100) is not in mixedCase
Parameter UserProfile.setFees(address)._a (contracts/mosaic-alpha-contracts/User/UserProfile.sol#106) is not in mixedCase
Parameter UserProfile.setAffiliate(address)._a (contracts/mosaic-alpha-contracts/User/UserProfile.sol#110) is not in mixedCase
Parameter UserProfile.setUserRank(address,uint8)._user (contracts/mosaic-alpha-contracts/User/UserProfile.sol#116) is not in mixedCase
Parameter UserProfile.setUserRank(address,uint8)._rank (contracts/mosaic-alpha-contracts/User/UserProfile.sol#116) is not in mixedCase
Parameter UserProfile.getParent(address)._user (contracts/mosaic-alpha-contracts/User/UserProfile.sol#133) is not in mixedCase
Parameter UserProfile.getParents(address)._user (contracts/mosaic-alpha-contracts/User/UserProfile.sol#138) is not in mixedCase
Parameter UserProfile.getParentsAndParentRanks(address)._user (contracts/mosaic-alpha-contracts/User/UserProfile.sol#143) is not in mixedCase
Parameter UserProfile.getParentsAndBuyFeeDiscount(address)._user (contracts/mosaic-alpha-contracts/User/UserProfile.sol#150) is not in mixedCase
Parameter UserProfile.getReferralCount(address)._user (contracts/mosaic-alpha-contracts/User/UserProfile.sol#159) is not in mixedCase
Parameter UserProfile.getActiveReferralCount(address)._user (contracts/mosaic-alpha-contracts/User/UserProfile.sol#164) is not in mixedCase
Parameter UserProfile.getAllReferrals(address)._user (contracts/mosaic-alpha-contracts/User/UserProfile.sol#169) is not in mixedCase
Parameter UserProfile.getReferrals(address,uint256,uint256)._user (contracts/mosaic-alpha-contracts/User/UserProfile.sol#174) is not in mixedCase
Parameter UserProfile.getReferrals(address,uint256,uint256)._index (contracts/mosaic-alpha-contracts/User/UserProfile.sol#174) is not in mixedCase
Parameter UserProfile.getReferrals(address,uint256,uint256)._count (contracts/mosaic-alpha-contracts/User/UserProfile.sol#174) is not in mixedCase
Parameter UserProfile.getUser(address)._user (contracts/mosaic-alpha-contracts/User/UserProfile.sol#191) is not in mixedCase
Parameter UserProfile.getUserRank(address)._user (contracts/mosaic-alpha-contracts/User/UserProfile.sol#196) is not in mixedCase
Parameter UserProfile.userExists(address)._user (contracts/mosaic-alpha-contracts/User/UserProfile.sol#205) is not in mixedCase
Parameter UserProfile.increaseActiveReferralCount(address)._user (contracts/mosaic-alpha-contracts/User/UserProfile.sol#209) is not in mixedCase
Parameter UserProfile.registerUser(address)._referredBy (contracts/mosaic-alpha-contracts/User/UserProfile.sol#214) is not in mixedCase
Parameter UserProfile.registerUser(address,address)._user (contracts/mosaic-alpha-contracts/User/UserProfile.sol#218) is not in mixedCase
Parameter UserProfile.registerUser(address,address)._referredBy (contracts/mosaic-alpha-contracts/User/UserProfile.sol#218) is not in mixedCase
Parameter UserProfile.registerUserWoBooster(address)._user (contracts/mosaic-alpha-contracts/User/UserProfile.sol#222) is not in mixedCase
Parameter Swaps.addTokens(address[])._tokens (contracts/mosaic-alpha-contracts/Vault/Swaps.sol#29) is not in mixedCase
Parameter Swaps.removeTokens(address[])._tokens (contracts/mosaic-alpha-contracts/Vault/Swaps.sol#40) is not in mixedCase
Parameter Swaps.quickQuoteMint(address,address,uint256)._poolAddr (contracts/mosaic-alpha-contracts/Vault/Swaps.sol#70) is not in mixedCase
Parameter Swaps.quickQuoteMint(address,address,uint256).LPTokensRequested (contracts/mosaic-alpha-contracts/Vault/Swaps.sol#70) is not in mixedCase
Parameter Swaps.quickQuoteBurn(address,address,uint256)._poolAddr (contracts/mosaic-alpha-contracts/Vault/Swaps.sol#91) is not in mixedCase
Parameter Swaps.quickQuoteBurn(address,address,uint256).LPTokensToBurn (contracts/mosaic-alpha-contracts/Vault/Swaps.sol#91) is not in mixedCase
Parameter Swaps.quoteMint(address,address[][],uint256)._poolAddr (contracts/mosaic-alpha-contracts/Vault/Swaps.sol#110) is not in mixedCase
Parameter Swaps.quoteMint(address,address[][],uint256)._routing (contracts/mosaic-alpha-contracts/Vault/Swaps.sol#110) is not in mixedCase
Parameter Swaps.quoteMint(address,address[][],uint256).LPTokensRequested (contracts/mosaic-alpha-contracts/Vault/Swaps.sol#110) is not in mixedCase
Parameter Swaps.quoteBurn(address,address[][],uint256)._poolAddr (contracts/mosaic-alpha-contracts/Vault/Swaps.sol#127) is not in mixedCase
Parameter Swaps.quoteBurn(address,address[][],uint256)._routing (contracts/mosaic-alpha-contracts/Vault/Swaps.sol#127) is not in mixedCase
Parameter Swaps.quoteBurn(address,address[][],uint256).LPTokensToBurn (contracts/mosaic-alpha-contracts/Vault/Swaps.sol#127) is not in mixedCase
Parameter Swaps.mint(address,address,address[][],uint256,uint256,address,uint256)._poolAddr (contracts/mosaic-alpha-contracts/Vault/Swaps.sol#144) is not in mixedCase
Parameter Swaps.mint(address,address,address[][],uint256,uint256,address,uint256)._routing (contracts/mosaic-alpha-contracts/Vault/Swaps.sol#144) is not in mixedCase
Parameter Swaps.burn(address,address,address[][],uint256,uint256,uint256)._poolAddr (contracts/mosaic-alpha-contracts/Vault/Swaps.sol#191) is not in mixedCase
Parameter Swaps.burn(address,address,address[][],uint256,uint256,uint256)._routing (contracts/mosaic-alpha-contracts/Vault/Swaps.sol#191) is not in mixedCase
Parameter Swaps.quickMint(address,address,uint256,uint256,address,uint256)._poolAddr (contracts/mosaic-alpha-contracts/Vault/Swaps.sol#222) is not in mixedCase
Parameter Swaps.quickBurn(address,address,uint256,uint256,uint256)._poolAddr (contracts/mosaic-alpha-contracts/Vault/Swaps.sol#257) is not in mixedCase
Parameter SystemRole.addSystemAdmin(address)._account (contracts/mosaic-alpha-contracts/helpers/SystemRole.sol#79) is not in mixedCase
Parameter SystemRole.removeSystemAdmin(address)._account (contracts/mosaic-alpha-contracts/helpers/SystemRole.sol#88) is not in mixedCase
Parameter SystemRole.addTrader(address)._account (contracts/mosaic-alpha-contracts/helpers/SystemRole.sol#97) is not in mixedCase
Parameter SystemRole.removeTrader(address)._account (contracts/mosaic-alpha-contracts/helpers/SystemRole.sol#106) is not in mixedCase
Parameter SystemRole.addInvestor(address)._account (contracts/mosaic-alpha-contracts/helpers/SystemRole.sol#115) is not in mixedCase
Parameter SystemRole.removeInvestor(address)._account (contracts/mosaic-alpha-contracts/helpers/SystemRole.sol#124) is not in mixedCase
Parameter SystemRole.addDepositLocker(address)._account (contracts/mosaic-alpha-contracts/helpers/SystemRole.sol#133) is not in mixedCase
Parameter SystemRole.removeDepositLocker(address)._account (contracts/mosaic-alpha-contracts/helpers/SystemRole.sol#142) is not in mixedCase
Parameter SystemRole.addAdmin(address)._account (contracts/mosaic-alpha-contracts/helpers/SystemRole.sol#151) is not in mixedCase
Parameter SystemRole.removeAdmin(address)._account (contracts/mosaic-alpha-contracts/helpers/SystemRole.sol#160) is not in mixedCase
Parameter SystemRole.checkSystemAdmin(address)._account (contracts/mosaic-alpha-contracts/helpers/SystemRole.sol#184) is not in mixedCase
Parameter SystemRole.checkTrader(address)._account (contracts/mosaic-alpha-contracts/helpers/SystemRole.sol#188) is not in mixedCase
Parameter SystemRole.checkInvestor(address)._account (contracts/mosaic-alpha-contracts/helpers/SystemRole.sol#192) is not in mixedCase
Parameter SystemRole.checkDepositLocker(address)._account (contracts/mosaic-alpha-contracts/helpers/SystemRole.sol#196) is not in mixedCase

```
Parameter SystemRole.checkAdmin(address)._account (contracts/mosaic-alpha-contracts/helpers/SystemRole.sol#200) is not in mixedCase
Parameter SystemWhitelist.isWhitelisted(address)._address (contracts/mosaic-alpha-contracts/helpers/SystemWhitelist.sol#48) is not in mixedCase
Parameter SystemWhitelist.isBlacklisted(address)._address (contracts/mosaic-alpha-contracts/helpers/SystemWhitelist.sol#57) is not in mixedCase
Parameter SystemWhitelist.addWhitelist(address)._address (contracts/mosaic-alpha-contracts/helpers/SystemWhitelist.sol#69) is not in mixedCase
Parameter SystemWhitelist.addWhitelistBulk(address[])._address (contracts/mosaic-alpha-contracts/helpers/SystemWhitelist.sol#81) is not in mixedCase
Parameter SystemWhitelist.addBlacklist(address)._address (contracts/mosaic-alpha-contracts/helpers/SystemWhitelist.sol#95) is not in mixedCase
Parameter SystemWhitelist.removeWhitelist(address)._address (contracts/mosaic-alpha-contracts/helpers/SystemWhitelist.sol#107) is not in mixedCase
Parameter SystemWhitelist.removeBlacklist(address)._address (contracts/mosaic-alpha-contracts/helpers/SystemWhitelist.sol#117) is not in mixedCase
Struct FixedPoint.uq112x112 (contracts/mosaic-alpha-contracts/helpers/pancake/libraries/FixedPoint.sol#12-14) is not in CapWords
Struct FixedPoint.uq144x112 (contracts/mosaic-alpha-contracts/helpers/pancake/libraries/FixedPoint.sol#18-20) is not in CapWords
Variable GovernedContractsMM.Running (contracts/test/ConfigGovernedExample.sol#12) is not in mixedCase
Variable GovernedContractsMM.address_slot (contracts/test/ConfigGovernedExample.sol#16) is not in mixedCase
Variable GovernedContractsMM.uint256_slot (contracts/test/ConfigGovernedExample.sol#17) is not in mixedCase
Variable GovernedContractsMM.bytes32_slot (contracts/test/ConfigGovernedExample.sol#18) is not in mixedCase
Modifier GovernedContractsMM.Live() (contracts/test/ConfigGovernedExample.sol#28-31) is not in mixedCase
Parameter TestToken.mint(uint256)._amount (contracts/test/TestToken.sol#12) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
```

variable-names-are-too-similar

```
Variable PancakePair.swap(uint256,uint256,address,bytes).balance0Adjusted (contracts/mosaic-alpha-contracts/helpers/pancake/PancakePair.sol#182) is too
similar to PancakePair.swap(uint256,uint256,address,bytes).balance1Adjusted (contracts/mosaic-alpha-contracts/helpers/pancake/PancakePair.sol#183)
Variable PancakePair.price0CumulativeLast (contracts/mosaic-alpha-contracts/helpers/pancake/PancakePair.sol#28) is too similar to
PancakePair.price1CumulativeLast (contracts/mosaic-alpha-contracts/helpers/pancake/PancakePair.sol#29)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar

Variable PancakeRouter.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (contracts/mosaic-alpha-
contracts/helpers/pancake/PancakeRouter.sol#66) is too similar to
PancakeRouter.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountBDesired (contracts/mosaic-alpha-
contracts/helpers/pancake/PancakeRouter.sol#67)
Variable IPancakeRouter01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (contracts/mosaic-alpha-
contracts/helpers/pancake/interfaces/IPancakeRouter01.sol#13) is too similar to
PancakeRouter._addLiquidity(address,address,uint256,uint256,uint256,uint256).amountBDesired (contracts/mosaic-alpha-
contracts/helpers/pancake/PancakeRouter.sol#39)
Variable IPancakeRouter01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (contracts/mosaic-alpha-
contracts/helpers/pancake/interfaces/IPancakeRouter01.sol#13) is too similar to
PancakeRouter.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountBDesired (contracts/mosaic-alpha-
contracts/helpers/pancake/PancakeRouter.sol#67)
Variable PancakeRouter._addLiquidity(address,address,uint256,uint256,uint256,uint256).amountADesired (contracts/mosaic-alpha-
contracts/helpers/pancake/PancakeRouter.sol#38) is too similar to
PancakeRouter._addLiquidity(address,address,uint256,uint256,uint256,uint256).amountBDesired (contracts/mosaic-alpha-
contracts/helpers/pancake/PancakeRouter.sol#39)
Variable IPancakeRouter01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (contracts/mosaic-alpha-
contracts/helpers/pancake/interfaces/IPancakeRouter01.sol#13) is too similar to
IPancakeRouter01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountBDesired (contracts/mosaic-alpha-
contracts/helpers/pancake/interfaces/IPancakeRouter01.sol#14)
Variable PancakeRouter._addLiquidity(address,address,uint256,uint256,uint256,uint256).amountADesired (contracts/mosaic-alpha-
contracts/helpers/pancake/PancakeRouter.sol#38) is too similar to
PancakeRouter.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountBDesired (contracts/mosaic-alpha-
contracts/helpers/pancake/PancakeRouter.sol#67)
Variable PancakeRouter.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (contracts/mosaic-alpha-
contracts/helpers/pancake/PancakeRouter.sol#66) is too similar to
PancakeRouter._addLiquidity(address,address,uint256,uint256,uint256,uint256).amountBDesired (contracts/mosaic-alpha-
contracts/helpers/pancake/PancakeRouter.sol#39)
Variable PancakeRouter._addLiquidity(address,address,uint256,uint256,uint256,uint256).amountADesired (contracts/mosaic-alpha-
contracts/helpers/pancake/PancakeRouter.sol#38) is too similar to
IPancakeRouter01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountBDesired (contracts/mosaic-alpha-
contracts/helpers/pancake/interfaces/IPancakeRouter01.sol#14)
Variable PancakeRouter.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (contracts/mosaic-alpha-
contracts/helpers/pancake/PancakeRouter.sol#66) is too similar to
IPancakeRouter01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountBDesired (contracts/mosaic-alpha-
contracts/helpers/pancake/interfaces/IPancakeRouter01.sol#14)
Variable PancakeRouter._addLiquidity(address,address,uint256,uint256,uint256,uint256).amountAOptimal (contracts/mosaic-alpha-
contracts/helpers/pancake/PancakeRouter.sol#56) is too similar to
PancakeRouter._addLiquidity(address,address,uint256,uint256,uint256,uint256).amountBOptimal (contracts/mosaic-alpha-
contracts/helpers/pancake/PancakeRouter.sol#51)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar

Variable Vault.emergencyWithdraw(address,address,bool)._tokenAmount (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1352) is too similar to
Vault._registerUserPurchase(address,address,address,address[],uint256[],uint256).tokenAmounts (contracts/mosaic-alpha-contracts/Vault/Vault.sol#205)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar

Variable Fees.allocatePerformanceFee(address,address,uint256,uint256,address).affiliateL1ClaimablePerLp (contracts/mosaic-alpha-
contracts/Fees/Fees.sol#513) is too similar to Fees.allocatePerformanceFee(address,address,uint256,uint256,address).affiliateL2ClaimablePerLp
(contracts/mosaic-alpha-contracts/Fees/Fees.sol#514)
Variable Fees.allocateTrailingFee(address,address,uint256,uint256,address).affiliateL1ClaimablePerLp (contracts/mosaic-alpha-contracts/Fees/Fees.sol#469)
is too similar to Fees.allocateTrailingFee(address,address,uint256,uint256,address).affiliateL2ClaimablePerLp (contracts/mosaic-alpha-
contracts/Fees/Fees.sol#470)
Variable Fees.allocatePerformanceFee(address,address,uint256,uint256,address).affiliateL1ClaimablePerLp (contracts/mosaic-alpha-
contracts/Fees/Fees.sol#513) is too similar to Fees.allocateTrailingFee(address,address,uint256,uint256,address).affiliateL2ClaimablePerLp
(contracts/mosaic-alpha-contracts/Fees/Fees.sol#470)
Variable Fees.allocateTrailingFee(address,address,uint256,uint256,address).affiliateL1ClaimablePerLp (contracts/mosaic-alpha-contracts/Fees/Fees.sol#469)
is too similar to Fees.allocatePerformanceFee(address,address,uint256,uint256,address).affiliateL2ClaimablePerLp (contracts/mosaic-alpha-
contracts/Fees/Fees.sol#514)
Variable Fees._calculateUserTimeIntegrals(address,uint32,uint8).level1xTime (contracts/mosaic-alpha-contracts/Fees/Fees.sol#170) is too similar to
Fees._calculateUserTimeIntegrals(address,uint32,uint8).level2xTime (contracts/mosaic-alpha-contracts/Fees/Fees.sol#170)
Variable Fees.traderFeeCurveX (contracts/mosaic-alpha-contracts/Fees/Fees.sol#53) is too similar to Fees.traderFeeCurveY (contracts/mosaic-alpha-
contracts/Fees/Fees.sol#52)
Variable Fees.claimUserClaimableFee(address,address).userClaimableL1 (contracts/mosaic-alpha-contracts/Fees/Fees.sol#216) is too similar to
Fees.claimUserClaimableFee(address,address).userClaimableL2 (contracts/mosaic-alpha-contracts/Fees/Fees.sol#217)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar

Variable SimpleDeposit.getAccountBalances(address)._accountBalance (contracts/mosaic-alpha-contracts/Deposit/SimpleDeposit.sol#241) is too similar to
SimpleDeposit.accountBalances (contracts/mosaic-alpha-contracts/Deposit/SimpleDeposit.sol#46)
Variable SimpleDeposit.getAccountBalance(address)._accountBalance (contracts/mosaic-alpha-contracts/Deposit/SimpleDeposit.sol#237) is too similar to
SimpleDeposit.accountBalances (contracts/mosaic-alpha-contracts/Deposit/SimpleDeposit.sol#46)
Variable ISimpleDeposit.getAccountBalances(address)._accountBalance (contracts/mosaic-alpha-contracts/Interfaces/ISimpleDeposit.sol#26) is too similar to
SimpleDeposit.accountBalances (contracts/mosaic-alpha-contracts/Deposit/SimpleDeposit.sol#46)
Variable SimpleDeposit.getAccountLockedBalance(address)._accountBalance (contracts/mosaic-alpha-contracts/Deposit/SimpleDeposit.sol#257) is too similar to
SimpleDeposit.accountBalances (contracts/mosaic-alpha-contracts/Deposit/SimpleDeposit.sol#46)
Variable SimpleDeposit.getExternalLockedBalance(address)._accountBalance (contracts/mosaic-alpha-contracts/Deposit/SimpleDeposit.sol#263) is too similar to
SimpleDeposit.accountBalances (contracts/mosaic-alpha-contracts/Deposit/SimpleDeposit.sol#46)
Variable PancakeOracleLibrary.currentCumulativePrices(address).price0Cumulative (contracts/mosaic-alpha-
contracts/helpers/pancake/libraries/PancakeOracleLibrary.sol#18) is too similar to PancakeOracleLibrary.currentCumulativePrices(address).price1Cumulative
(contracts/mosaic-alpha-contracts/helpers/pancake/libraries/PancakeOracleLibrary.sol#18)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar
```

too-many-digits

```
PancakeFactory.createPair(address,address) (contracts/mosaic-alpha-contracts/helpers/pancake/PancakeFactory.sol#29-44) uses literals with too many digits:
    - bytecode = type(address)(PancakePair).creationCode (contracts/mosaic-alpha-contracts/helpers/pancake/PancakeFactory.sol#34)
PancakeFactory.slitherConstructorConstantVariables() (contracts/mosaic-alpha-contracts/helpers/pancake/PancakeFactory.sol#10-56) uses literals with too
many digits:
    - INIT_CODE_PAIR_HASH = keccak256(bytes)(abi.encodePacked(type(address)(PancakePair).creationCode)) (contracts/mosaic-alpha-
```

```
contracts/helpers/pancake/PancakeFactory.sol#11)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits

FixedPoint.slitherConstructorConstantVariables() (contracts/mosaic-alpha-contracts/Pool/Math.sol#4-609) uses literals with too many digits:
    - x0 = 128000000000000000000 (contracts/mosaic-alpha-contracts/Pool/Math.sol#156)
FixedPoint.slitherConstructorConstantVariables() (contracts/mosaic-alpha-contracts/Pool/Math.sol#4-609) uses literals with too many digits:
    - a0 = 38877084059945950922200000000000000000000000000000000000 (contracts/mosaic-alpha-contracts/Pool/Math.sol#157)
FixedPoint.slitherConstructorConstantVariables() (contracts/mosaic-alpha-contracts/Pool/Math.sol#4-609) uses literals with too many digits:
    - x1 = 64000000000000000000 (contracts/mosaic-alpha-contracts/Pool/Math.sol#158)
FixedPoint.slitherConstructorConstantVariables() (contracts/mosaic-alpha-contracts/Pool/Math.sol#4-609) uses literals with too many digits:
    - a1 = 6235149080811616882910000000 (contracts/mosaic-alpha-contracts/Pool/Math.sol#159)
FixedPoint.slitherConstructorConstantVariables() (contracts/mosaic-alpha-contracts/Pool/Math.sol#4-609) uses literals with too many digits:
    - x2 = 3200000000000000000000 (contracts/mosaic-alpha-contracts/Pool/Math.sol#162)
FixedPoint.slitherConstructorConstantVariables() (contracts/mosaic-alpha-contracts/Pool/Math.sol#4-609) uses literals with too many digits:
    - a2 = 7896296018268069516100000000000000 (contracts/mosaic-alpha-contracts/Pool/Math.sol#163)
FixedPoint.slitherConstructorConstantVariables() (contracts/mosaic-alpha-contracts/Pool/Math.sol#4-609) uses literals with too many digits:
    - x3 = 160000000000000000000 (contracts/mosaic-alpha-contracts/Pool/Math.sol#164)
FixedPoint.slitherConstructorConstantVariables() (contracts/mosaic-alpha-contracts/Pool/Math.sol#4-609) uses literals with too many digits:
    - a3 = 888611052050787263676000000 (contracts/mosaic-alpha-contracts/Pool/Math.sol#165)
FixedPoint.slitherConstructorConstantVariables() (contracts/mosaic-alpha-contracts/Pool/Math.sol#4-609) uses literals with too many digits:
    - x4 = 800000000000000000000 (contracts/mosaic-alpha-contracts/Pool/Math.sol#166)
FixedPoint.slitherConstructorConstantVariables() (contracts/mosaic-alpha-contracts/Pool/Math.sol#4-609) uses literals with too many digits:
    - x5 = 4000000000000000000000 (contracts/mosaic-alpha-contracts/Pool/Math.sol#168)
FixedPoint.slitherConstructorConstantVariables() (contracts/mosaic-alpha-contracts/Pool/Math.sol#4-609) uses literals with too many digits:
    - x6 = 20000000000000000000000 (contracts/mosaic-alpha-contracts/Pool/Math.sol#170)
FixedPoint.slitherConstructorConstantVariables() (contracts/mosaic-alpha-contracts/Pool/Math.sol#4-609) uses literals with too many digits:
    - x7 = 100000000000000000000 (contracts/mosaic-alpha-contracts/Pool/Math.sol#172)
FixedPoint.slitherConstructorConstantVariables() (contracts/mosaic-alpha-contracts/Pool/Math.sol#4-609) uses literals with too many digits:
    - x8 = 500000000000000000000 (contracts/mosaic-alpha-contracts/Pool/Math.sol#174)
FixedPoint.slitherConstructorConstantVariables() (contracts/mosaic-alpha-contracts/Pool/Math.sol#4-609) uses literals with too many digits:
    - x9 = 2500000000000000000000 (contracts/mosaic-alpha-contracts/Pool/Math.sol#176)
FixedPoint.slitherConstructorConstantVariables() (contracts/mosaic-alpha-contracts/Pool/Math.sol#4-609) uses literals with too many digits:
    - x10 = 12500000000000000000000 (contracts/mosaic-alpha-contracts/Pool/Math.sol#178)
FixedPoint.slitherConstructorConstantVariables() (contracts/mosaic-alpha-contracts/Pool/Math.sol#4-609) uses literals with too many digits:
    - x11 = 62500000000000000000000 (contracts/mosaic-alpha-contracts/Pool/Math.sol#180)
console.slitherConstructorConstantVariables() (node_modules/hardhat/console.sol#4-1532) uses literals with too many digits:
    - CONSOLE_ADDRESS = address(0x000000000000000000636F6e736F6c652e6c6f67) (node_modules/hardhat/console.sol#5)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits

PoolFactory.create(string,string,uint8,address[],uint32[],uint256[],IFees.MosaicPoolFees,address,bool) (contracts/mosaic-alpha-
contracts/Pool/PoolFactory.sol#120-263) uses literals with too many digits:
    - mstore(uint256,uint256)(0x0,0x5880730000000000000000000000000000000000000000803b80938091923cF3 | source * 0x100000000000000000) (contracts/mosaic-
alpha-contracts/Pool/PoolFactory.sol#166-172)
Babylonian.sqrt(uint256) (contracts/mosaic-alpha-contracts/helpers/pancake/libraries/Babylonian.sol#10-52) uses literals with too many digits:
    - xx >= 0x100000000000000000000000000000000 (contracts/mosaic-alpha-contracts/helpers/pancake/libraries/Babylonian.sol#16)
Babylonian.sqrt(uint256) (contracts/mosaic-alpha-contracts/helpers/pancake/libraries/Babylonian.sol#10-52) uses literals with too many digits:
    - xx >= 0x10000000000000000 (contracts/mosaic-alpha-contracts/helpers/pancake/libraries/Babylonian.sol#20)
Babylonian.sqrt(uint256) (contracts/mosaic-alpha-contracts/helpers/pancake/libraries/Babylonian.sol#10-52) uses literals with too many digits:
    - xx >= 0x100000000 (contracts/mosaic-alpha-contracts/helpers/pancake/libraries/Babylonian.sol#24)
BitMath.mostSignificantBit(uint256) (contracts/mosaic-alpha-contracts/helpers/pancake/libraries/BitMath.sol#7-39) uses literals with too many digits:
    - x >= 0x100000000000000000000000000000000 (contracts/mosaic-alpha-contracts/helpers/pancake/libraries/BitMath.sol#10)
BitMath.mostSignificantBit(uint256) (contracts/mosaic-alpha-contracts/helpers/pancake/libraries/BitMath.sol#7-39) uses literals with too many digits:
    - x >= 0x10000000000000000 (contracts/mosaic-alpha-contracts/helpers/pancake/libraries/BitMath.sol#14)
BitMath.mostSignificantBit(uint256) (contracts/mosaic-alpha-contracts/helpers/pancake/libraries/BitMath.sol#7-39) uses literals with too many digits:
    - x >= 0x100000000 (contracts/mosaic-alpha-contracts/helpers/pancake/libraries/BitMath.sol#18)
FixedPoint.slitherConstructorConstantVariables() (contracts/mosaic-alpha-contracts/helpers/pancake/libraries/FixedPoint.sol#9-147) uses literals with too
many digits:
    - Q112 = 0x10000000000000000000000000000000 (contracts/mosaic-alpha-contracts/helpers/pancake/libraries/FixedPoint.sol#23)
FixedPoint.slitherConstructorConstantVariables() (contracts/mosaic-alpha-contracts/helpers/pancake/libraries/FixedPoint.sol#9-147) uses literals with too
many digits:
    - Q224 = 0x100000000000000000000000000000000000000000000000000000000000000000 (contracts/mosaic-alpha-contracts/helpers/pancake/libraries/FixedPoint.sol#24)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
```

unused-state-variable

```
FixedPoint.MIN_POW_BASE_FREE_EXPONENT (contracts/mosaic-alpha-contracts/Pool/Math.sol#11) is never used in FixedPoint (contracts/mosaic-alpha-
contracts/Pool/Math.sol#4-609)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variable

Initializable._____gap (contracts/mosaic-alpha-contracts/Oracle/Initializable.sol#66) is never used in SimpleMosaicOracle (contracts/mosaic-alpha-
contracts/Oracle/SimpleMosaicOracle.sol#20-394)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variable
```

public-function-that-could-be-declared-external

```
quote(uint256,uint256,uint256) should be declared external:
    - PancakeRouter.quote(uint256,uint256,uint256) (contracts/mosaic-alpha-contracts/helpers/pancake/PancakeRouter.sol#405-407)
getAmountOut(uint256,uint256,uint256) should be declared external:
    - PancakeRouter.getAmountOut(uint256,uint256,uint256) (contracts/mosaic-alpha-contracts/helpers/pancake/PancakeRouter.sol#409-417)
getAmountIn(uint256,uint256,uint256) should be declared external:
    - PancakeRouter.getAmountIn(uint256,uint256,uint256) (contracts/mosaic-alpha-contracts/helpers/pancake/PancakeRouter.sol#419-427)
getAmountsOut(uint256,address[]) should be declared external:
    - PancakeRouter.getAmountsOut(uint256,address[]) (contracts/mosaic-alpha-contracts/helpers/pancake/PancakeRouter.sol#429-437)
getAmountsIn(uint256,address[]) should be declared external:
    - PancakeRouter.getAmountsIn(uint256,address[]) (contracts/mosaic-alpha-contracts/helpers/pancake/PancakeRouter.sol#439-447)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external

owner() should be declared external:
    - Pool.owner() (contracts/mosaic-alpha-contracts/Pool/Pool.sol#114-116)
setManagers(address[]) should be declared external:
    - Pool.setManagers(address[]) (contracts/mosaic-alpha-contracts/Pool/Pool.sol#268-271)
calcMintAmounts(uint256) should be declared external:
    - Pool.calcMintAmounts(uint256) (contracts/mosaic-alpha-contracts/Pool/Pool.sol#334-336)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external

setToDeployed() should be declared external:
    - Governance.setToDeployed() (contracts/mosaic-alpha-contracts/Governance/Governance.sol#122-126)
ActivateDeployedMosaic(address,address,address,address,address,address,address,address,address,address,address) should be declared external:
    - Governance.ActivateDeployedMosaic(address,address,address,address,address,address,address,address,address,address,address) (contracts/mosaic-alpha-
contracts/Governance/Governance.sol#128-171)
read_core_Running() should be declared external:
    - Governance.read_core_Running() (contracts/mosaic-alpha-contracts/Governance/Governance.sol#321)
read_core_govAddr() should be declared external:
    - Governance.read_core_govAddr() (contracts/mosaic-alpha-contracts/Governance/Governance.sol#322)
read_core_owners() should be declared external:
    - Governance.read_core_owners() (contracts/mosaic-alpha-contracts/Governance/Governance.sol#324)
read_config_Main_addressN(uint256) should be declared external:
    - Governance.read_config_Main_addressN(uint256) (contracts/mosaic-alpha-contracts/Governance/Governance.sol#326-328)
read_config_core(string) should be declared external:
    - Governance.read_config_core(string) (contracts/mosaic-alpha-contracts/Governance/Governance.sol#331-340)
read_config_name(string) should be declared external:
    - Governance.read_config_name(string) (contracts/mosaic-alpha-contracts/Governance/Governance.sol#341)
read_config_emergencyStatus(string) should be declared external:
    - Governance.read_config_emergencyStatus(string) (contracts/mosaic-alpha-contracts/Governance/Governance.sol#342)
```

```
read_config_governAddress(string) should be declared external:
    - Governance.read_config_governAddress(string) (contracts/mosaic-alpha-contracts/Governance/Governance.sol#343)
read_config_Managers(string) should be declared external:
    - Governance.read_config_Managers(string) (contracts/mosaic-alpha-contracts/Governance/Governance.sol#344)
read_config_bool_slot(string) should be declared external:
    - Governance.read_config_bool_slot(string) (contracts/mosaic-alpha-contracts/Governance/Governance.sol#346)
read_config_address_slot(string) should be declared external:
    - Governance.read_config_address_slot(string) (contracts/mosaic-alpha-contracts/Governance/Governance.sol#347)
read_config_uint256_slot(string) should be declared external:
    - Governance.read_config_uint256_slot(string) (contracts/mosaic-alpha-contracts/Governance/Governance.sol#348)
read_config_bytes32_slot(string) should be declared external:
    - Governance.read_config_bytes32_slot(string) (contracts/mosaic-alpha-contracts/Governance/Governance.sol#349)
read_config_Managers_batched(string,uint256[]) should be declared external:
    - Governance.read_config_Managers_batched(string,uint256[]) (contracts/mosaic-alpha-contracts/Governance/Governance.sol#351-357)
read_config_bool_slot_batched(string,uint256[]) should be declared external:
    - Governance.read_config_bool_slot_batched(string,uint256[]) (contracts/mosaic-alpha-contracts/Governance/Governance.sol#359-365)
read_config_address_slot_batched(string,uint256[]) should be declared external:
    - Governance.read_config_address_slot_batched(string,uint256[]) (contracts/mosaic-alpha-contracts/Governance/Governance.sol#367-373)
read_config_uint256_slot_batched(string,uint256[]) should be declared external:
    - Governance.read_config_uint256_slot_batched(string,uint256[]) (contracts/mosaic-alpha-contracts/Governance/Governance.sol#375-381)
read_config_bytes32_slot_batched(string,uint256[]) should be declared external:
    - Governance.read_config_bytes32_slot_batched(string,uint256[]) (contracts/mosaic-alpha-contracts/Governance/Governance.sol#383-389)
read_invoteConfig_core(string) should be declared external:
    - Governance.read_invoteConfig_core(string) (contracts/mosaic-alpha-contracts/Governance/Governance.sol#394-403)
read_invoteConfig_name(string) should be declared external:
    - Governance.read_invoteConfig_name(string) (contracts/mosaic-alpha-contracts/Governance/Governance.sol#404)
read_invoteConfig_emergencyStatus(string) should be declared external:
    - Governance.read_invoteConfig_emergencyStatus(string) (contracts/mosaic-alpha-contracts/Governance/Governance.sol#405)
read_invoteConfig_governAddress(string) should be declared external:
    - Governance.read_invoteConfig_governAddress(string) (contracts/mosaic-alpha-contracts/Governance/Governance.sol#406)
read_invoteConfig_Managers(string) should be declared external:
    - Governance.read_invoteConfig_Managers(string) (contracts/mosaic-alpha-contracts/Governance/Governance.sol#407)
read_invoteConfig_boolslot(string) should be declared external:
    - Governance.read_invoteConfig_boolslot(string) (contracts/mosaic-alpha-contracts/Governance/Governance.sol#408)
read_invoteConfig_address_slot(string) should be declared external:
    - Governance.read_invoteConfig_address_slot(string) (contracts/mosaic-alpha-contracts/Governance/Governance.sol#409)
read_invoteConfig_uint256_slot(string) should be declared external:
    - Governance.read_invoteConfig_uint256_slot(string) (contracts/mosaic-alpha-contracts/Governance/Governance.sol#410)
read_invoteConfig_bytes32_slot(string) should be declared external:
    - Governance.read_invoteConfig_bytes32_slot(string) (contracts/mosaic-alpha-contracts/Governance/Governance.sol#411)
generator(string) should be declared external:
    - Governance.generator(string) (contracts/mosaic-alpha-contracts/Governance/Governance.sol#457-460)
getGovernanceState() should be declared external:
    - Governed.getGovernanceState() (contracts/mosaic-alpha-contracts/Governance/Governed.sol#25-30)
getBidPriceAt(uint256,uint32) should be declared external:
    - Vault.getBidPriceAt(uint256,uint32) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1122-1124)
getBidPrice(uint256) should be declared external:
    - Vault.getBidPrice(uint256) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1126-1128)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external

allocateBuyFee(address,address,address,uint256) should be declared external:
    - Fees.allocateBuyFee(address,address,address,uint256) (contracts/mosaic-alpha-contracts/Fees/Fees.sol#417-446)
isValidPoolFees(IFees.MosaicPoolFees) should be declared external:
    - Fees.isValidPoolFees(IFees.MosaicPoolFees) (contracts/mosaic-alpha-contracts/Fees/Fees.sol#798-805)
calculateBuyFeeDistribution(address,address,uint256,uint16) should be declared external:
    - Fees.calculateBuyFeeDistribution(address,address,uint256,uint16) (contracts/mosaic-alpha-contracts/Fees/Fees.sol#822-824)
isController(address) should be declared external:
    - Controllable.isController(address) (contracts/mosaic-alpha-contracts/helpers/Controllable.sol#49-51)
renounceOwnership() should be declared external:
    - Ownable.renounceOwnership() (contracts/mosaic-alpha-contracts/helpers/Ownable.sol#59-61)
transferOwnership(address) should be declared external:
    - Ownable.transferOwnership(address) (contracts/mosaic-alpha-contracts/helpers/Ownable.sol#67-70)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external

grantRole(bytes32,address) should be declared external:
    - AccessControl.grantRole(bytes32,address) (node_modules/@openzeppelin/contracts/access/AccessControl.sol#144-146)
renounceRole(bytes32,address) should be declared external:
    - AccessControl.renounceRole(bytes32,address) (node_modules/@openzeppelin/contracts/access/AccessControl.sol#179-183)
transferOwnership(address) should be declared external:
    - Ownable.transferOwnership(address) (node_modules/@openzeppelin/contracts/access/Ownable.sol#69-72)
    - SimpleDeposit.transferOwnership(address) (contracts/mosaic-alpha-contracts/Deposit/SimpleDeposit.sol#82-87)
    - SystemRole.transferOwnership(address) (contracts/mosaic-alpha-contracts/helpers/SystemRole.sol#70-72)
    - SystemWhitelist.transferOwnership(address) (contracts/mosaic-alpha-contracts/helpers/SystemWhitelist.sol#39-41)
name() should be declared external:
    - ERC20.name() (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#62-64)
symbol() should be declared external:
    - ERC20.symbol() (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#70-72)
decimals() should be declared external:
    - ERC20.decimals() (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#87-89)
totalSupply() should be declared external:
    - ERC20.totalSupply() (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#94-96)
balanceOf(address) should be declared external:
    - ERC20.balanceOf(address) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#101-103)
transfer(address,uint256) should be declared external:
    - ERC20.transfer(address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#113-117)
approve(address,uint256) should be declared external:
    - ERC20.approve(address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#136-140)
transferFrom(address,address,uint256) should be declared external:
    - ERC20.transferFrom(address,address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#158-167)
increaseAllowance(address,uint256) should be declared external:
    - ERC20.increaseAllowance(address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#181-185)
decreaseAllowance(address,uint256) should be declared external:
    - ERC20.decreaseAllowance(address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#201-210)
burn(uint256) should be declared external:
    - ERC20Burnable.burn(uint256) (node_modules/@openzeppelin/contracts/token/ERC20/extensions/ERC20Burnable.sol#20-22)
burnFrom(address,uint256) should be declared external:
    - ERC20Burnable.burnFrom(address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/extensions/ERC20Burnable.sol#35-38)
removeBlacklist(address) should be declared external:
    - Register.removeBlacklist(address) (contracts/mosaic-alpha-contracts/Register/Register.sol#178-182)
removeSystemAdmin(address) should be declared external:
    - SystemRole.removeSystemAdmin(address) (contracts/mosaic-alpha-contracts/helpers/SystemRole.sol#88-90)
addTrader(address) should be declared external:
    - SystemRole.addTrader(address) (contracts/mosaic-alpha-contracts/helpers/SystemRole.sol#97-99)
removeTrader(address) should be declared external:
    - SystemRole.removeTrader(address) (contracts/mosaic-alpha-contracts/helpers/SystemRole.sol#106-108)
addInvestor(address) should be declared external:
    - SystemRole.addInvestor(address) (contracts/mosaic-alpha-contracts/helpers/SystemRole.sol#115-117)
removeInvestor(address) should be declared external:
    - SystemRole.removeInvestor(address) (contracts/mosaic-alpha-contracts/helpers/SystemRole.sol#124-126)
addDepositLocker(address) should be declared external:
    - SystemRole.addDepositLocker(address) (contracts/mosaic-alpha-contracts/helpers/SystemRole.sol#133-135)
removeDepositLocker(address) should be declared external:
    - SystemRole.removeDepositLocker(address) (contracts/mosaic-alpha-contracts/helpers/SystemRole.sol#142-144)
removeAdmin(address) should be declared external:
    - SystemRole.removeAdmin(address) (contracts/mosaic-alpha-contracts/helpers/SystemRole.sol#160-162)
removeBlacklist(address) should be declared external:
    - SystemWhitelist.removeBlacklist(address) (contracts/mosaic-alpha-contracts/helpers/SystemWhitelist.sol#117-121)
```

```
selfManageMe() should be declared external:
    - GovernedContractsMM.selfManageMe() (contracts/test/ConfigGovernedExample.sol#43-52)
tester() should be declared external:
    - GovernedContractsMM.tester() (contracts/test/ConfigGovernedExample.sol#56-59)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
. analyzed (169 contracts with 78 detectors), 1400 result(s) found
```

Authorization Report

Mosaic Alpha Smart Contract Analysis Appendix C

Suspicious authorization

Function name	Verdict
`Swaps.addTokens` and `Swaps.removeTokens`	**Faulty.** Can cause inconveniences, but can be replaced if needed. See Issue #1 in the report.
`SimpleAffiliateBoosterController.getTicketPrice`	**Faulty**, but the contract is not deployed. See Issue #2 in the report.
`Pool.updatePerfFeePricePerLp`	**Faulty.** The contract can be updated, so new pools won't have the issue, but already instantiated Pools can't. See Issue #3 in the report.
`Pool.initialize`	**Intentional.** Proxy, called by the factory.
`Pool.updateMinDuration`	**Intentional**, only uses data coming from the Vault.
`Fees.claimUserClaimableFee`	**Intentional**: it is part of the game, that you can claim instead of someone for a small commission.
`Fees.claimTraderClaimableFee`	**Intentional**: it is part of the game, that you can claim instead of someone for a small commission.
`Vault.disableFees`	**Intentional.**
`Vault.Burn`	**Intentional.**

Access Control

└	Function Name	Visibility	Mutability	Modifiers	Access Control
SimpleDeposit	Implementation	Ownable, SystemRole, ReentrancyGuard, ISimpleDeposit, Governed			
└		Public ❗	🔴	Ownable	
└	_selfManageMeBefore	Internal 🔒	🔴		
└	_selfManageMeAfter	Internal 🔒	🔴		
└	_onBeforeEmergencyChange	Internal 🔒	🔴		
└	transferOwnership	Public ❗	🔴	Live onlyOwner	onlyOwner
└	changeRegister	External ❗	🔴	Live onlyOwner	onlyOwner
└	changeFees	External ❗	🔴	Live onlyOwner	onlyOwner
└	changeAffiliate	External ❗	🔴	Live onlyOwner	onlyOwner
└	setWhitelistingEnabled	External ❗	🔴	Live onlyOwner	onlyOwner
└	setLockTime	External ❗	🔴	onlyOwner Live	onlyOwner
└	_handleUserStakeChanged	Internal 🔒	🔴		
└	approveExternalAddress	External ❗	🔴	Live	
└	disapproveExternalAddress	External ❗	🔴	Live	
└	isApprovedExternalAddress	External ❗		NO ❗	
└	_checkExternalAddress	Internal 🔒			
└	deposit	External ❗	🔴	nonReentrant Live	
└	depositTo	External ❗	🔴	nonReentrant Live	
└	withdraw	External ❗	🔴	nonReentrant Live	
└	userLock	External ❗	🔴	nonReentrant Live	
└	userLock	External ❗	🔴	nonReentrant Live	
└	userUnlock	External ❗	🔴	nonReentrant Live	
└	depositAndLock	External ❗	🔴	nonReentrant Live	
└	depositAndLock	External ❗	🔴	nonReentrant Live	
└	unlockAndWithdraw	External ❗	🔴	nonReentrant Live	
└	externalLock	External ❗	🔴	nonReentrant onlyDepositLocker Live	onlyDepositLocker
└	externalUnlock	External ❗	🔴	nonReentrant onlyDepositLocker Live	onlyDepositLocker
└	getLockTime	External ❗		NO ❗	
└	getAccountBalance	External ❗		NO ❗	
└	getAccountBalances	External ❗		NO ❗	
└	getAccountLockedBalanceUnlockTimestamp	External ❗		NO ❗	
└	getWithdrawableAccountLockedBalance	External ❗		NO ❗	
└	getAccountLockedBalance	External ❗		NO ❗	
└	getExternalLockedBalance	External ❗		NO ❗	
└	_deposit	Internal 🔒	🔴		
└	_withdraw	Internal 🔒	🔴		
└	_userLock	Internal 🔒	🔴		
└	_userUnlock	Internal 🔒	🔴		
└	_externalLock	Internal 🔒	🔴		
└	_externalUnlock	Internal 🔒	🔴		
FeeBurner	Implementation	IFeeBurner, Ownable, Governed			
└		Public ❗	🔴	NO ❗	
└	_selfManageMeBefore	Internal 🔒	🔴		
└	_selfManageMeAfter	Internal 🔒	🔴		
└	_onBeforeEmergencyChange	Internal 🔒	🔴		
└	burn	External ❗	🔴	Live	
└	setFeeToken	External ❗	🔴	Live onlyOwner	onlyOwner
└	setFeeTokenBurnable	External ❗	🔴	Live onlyOwner	onlyOwner
└	setFeeTo	External ❗	🔴	Live onlyOwner	onlyOwner
Fees	Implementation	IFees, Ownable, Governed			
└	getFeeCurve	External ❗		NO ❗	
└	getTraderFeeCurve	External ❗		NO ❗	
└	getUserPoolFeeStatus	External ❗		NO ❗	
└	getPoolFeeStatus	External ❗		NO ❗	
└		Public ❗	🔴	NO ❗	
└	selfManageMe	External ❗	🔴	NO ❗	onlyManagersFun
└	_selfManageMeBefore	Internal 🔒	🔴		
└	_selfManageMeAfter	Internal 🔒	🔴		
└	_onBeforeEmergencyChange	Internal 🔒	🔴		
└	_calculateUserTimeIntegrals	Internal 🔒	🔴		
└	_updateUserTimeIntegrals	Internal 🔒	🔴		
└	_getCappedAffiliateAmount	Internal 🔒	🔴		

└	Function Name	Visibility	Mutability	Modifiers	Access Control
└	claimUserClaimableFee	External ❗	🔴	Live	?
└	claimTraderClaimableFee	External ❗	🔴	Live	?
└	_calculateAffiliateRewardDistribution	Internal 🔒			
└	_allocateUserClaimableFees	Internal 🔒	🔴		
└	_allocateUserL1ClaimableFees	Internal 🔒	🔴		
└	_allocateUserL2ClaimableFees	Internal 🔒	🔴		
└	allocateBuyFee	Public ❗	🔴	Live	msg.sender == vault
└	allocateTrailingFee	Public ❗	🔴	Live	msg.sender == vault
└	allocatePerformanceFee	Public ❗	🔴	Live	msg.sender == vault
└	onBeforeTransfer	External ❗	🔴	NO ❗	msg.sender == vault
└	_getParentsFromCache	Internal 🔒	🔴		
└	updateDiscountCurve	External ❗	🔴	Live onlyOwner	onlyOwner
└	_updateDiscountCurve	Internal 🔒	🔴	Live onlyOwner	
└	updateTraderShareCurve	External ❗	🔴	Live onlyOwner	onlyOwner
└	_updateTraderShareCurve	Internal 🔒	🔴		
└	setExecutorShares	External ❗	🔴	Live onlyOwner	onlyOwner
└	setTraderClaimer	External ❗	🔴	Live	
└	updateFeeRanges	External ❗	🔴	Live onlyOwner	onlyOwner
└	getFeeRanges	External ❗		NO ❗	
└	getUserFeeLevels	External ❗		NO ❗	
└	updateFeeDistribution	External ❗	🔴	Live onlyOwner	onlyOwner
└	getFeeDistribution	External ❗		NO ❗	
└	updatePlatformFeeShares	External ❗	🔴	Live onlyOwner	onlyOwner
└	getPlatformFeeShares	External ❗		NO ❗	
└	isValidFeeRanges	Public ❗		NO ❗	
└	isValidFeeDistribution	Public ❗		NO ❗	
└	isValidPoolFees	Public ❗		NO ❗	
└	isValidBuyFee	External ❗		NO ❗	
└	isValidTrailingFee	External ❗		NO ❗	
└	isValidPerformanceFee	External ❗		NO ❗	
└	calculateBuyFeeDistribution	Public ❗		NO ❗	
└	_calculateBuyFeeDistribution	Internal 🔒			
└	_calculateEffectiveBuyFeeDiscountAmount	Public ❗		NO ❗	
└	calculateTrailingFeeTraderDistribution	External ❗		NO ❗	
└	_calculateEffectiveTraderRevenueShareAmount	Public ❗		NO ❗	
└	calculateTraderFeeDistribution	External ❗		NO ❗	
└	userRankChanged	External ❗	🔴	Live	msg.sender == userProfile
└	userStakeChanged	External ❗	🔴	Live	msg.sender == deposit
└	getUserFeeDiscountLevel	External ❗		NO ❗	
└	getTraderRevenueShareLevel	External ❗		NO ❗	
Governance	Implementation				
└		Public ❗	🔴	NO ❗	
└	core_govAddr_conf	Private 🔐	🔴		
└	core_emergency_conf	Private 🔐	🔴		
└	core_managers_conf	Private 🔐	🔴		
└	onlyManagers	Internal 🔒			
└	setToDeployed	Public ❗	🔴	NO ❗	onlyManagers
└	ActivateDeployedMosaic	Public ❗	🔴	NO ❗	onlyManagers
└	transfer_proportion	External ❗	🔴	NO ❗	curator_proportions[msg.sender] >=
└	update_config	Internal 🔒	🔴		
└	votein_config	Internal 🔒	🔴		
└	propose_config	External ❗	🔴	NO ❗	curator_proportions[msg.sender] >=
└	propose_core_change	External ❗	🔴	NO ❗	curator_proportions[msg.sender] >=
└	support_config_proposal	External ❗	🔴	NO ❗	curator_proportions[msg.sender] >=
└	read_core_Running	Public ❗		NO ❗	
└	read_core_govAddr	Public ❗		NO ❗	
└	read_core_managers	Public ❗		NO ❗	
└	read_core_owners	Public ❗		NO ❗	
└	read_config_Main_addressN	Public ❗		NO ❗	
└	read_config_core	Public ❗		NO ❗	
└	read_config_name	Public ❗		NO ❗	
└	read_config_emergencyStatus	Public ❗		NO ❗	
└	read_config_governAddress	Public ❗		NO ❗	
└	read_config_Managers	Public ❗		NO ❗	
└	read_config_bool_slot	Public ❗		NO ❗	
└	read_config_address_slot	Public ❗		NO ❗	
└	read_config_uint256_slot	Public ❗		NO ❗	
└	read_config_bytes32_slot	Public ❗		NO ❗	

└	Function Name	Visibility	Mutability	Modifiers	Access Control
└	read_config_Managers_batched	Public ❗		NO ❗	
└	read_config_bool_slot_batched	Public ❗		NO ❗	
└	read_config_address_slot_batched	Public ❗		NO ❗	
└	read_config_uint256_slot_batched	Public ❗		NO ❗	
└	read_config_bytes32_slot_batched	Public ❗		NO ❗	
└	read_invoteConfig_core	Public ❗		NO ❗	
└	read_invoteConfig_name	Public ❗		NO ❗	
└	read_invoteConfig_emergencyStatus	Public ❗		NO ❗	
└	read_invoteConfig_governAddress	Public ❗		NO ❗	
└	read_invoteConfig_Managers	Public ❗		NO ❗	
└	read_invoteConfig_boolslot	Public ❗		NO ❗	
└	read_invoteConfig_address_slot	Public ❗		NO ❗	
└	read_invoteConfig_uint256_slot	Public ❗		NO ❗	
└	read_invoteConfig_bytes32_slot	Public ❗		NO ❗	
└	propose_action	External ❗	🔴	NO ❗	curator_proportions[msg.sender] >=
└	support_actions	External ❗	🔴	NO ❗	curator_proportions[msg.sender] >=
└	trigger_action	External ❗	🔴	NO ❗	action_can_be_triggered_by[_id] == msg.sender
└	generator	Public ❗		NO ❗	
└	update_All	External ❗	🔴	onlyCurators	onlyCurators
└	selfManageMe_All	External ❗	🔴	onlyCurators	onlyCurators
└	execute_Manage	External ❗	🔴	onlyCurators	onlyCurators
└	_execute_Manage	Internal 🔒	🔴		
└	execute_batch_Manage	External ❗	🔴	onlyCurators	onlyCurators
└	execute_ManageBytes	External ❗	🔴	onlyCurators	onlyCurators
└	execute_batch_ManageBytes	External ❗	🔴	onlyCurators	onlyCurators
└	_execute_ManageBytes	Internal 🔒	🔴		
└	execute_ManageList	External ❗	🔴	onlyCurators	onlyCurators
└	execute_Vault_update	External ❗	🔴	onlyCurators	onlyCurators
└	_evaluateCallReturn	Internal 🔒			
Governed	Implementation	IGoverned			
└		Public ❗	🔴	NO ❗	
└	getGovernanceState	Public ❗		NO ❗	
└	LiveFun	Internal 🔒			
└	notLiveFun	Internal 🔒			
└	onlyManagersFun	Internal 🔒			
└	isManagerFun	Internal 🔒			
└	selfManageMe	External ❗	🔴	NO ❗	onlyManagersFun
└	_selfManageMeBefore	Internal 🔒	🔴		
└	_selfManageMeAfter	Internal 🔒	🔴		
└	_onBeforeEmergencyChange	Internal 🔒	🔴		
IERC3156FlashBorrower	Interface				
└	onFlashLoan	External ❗	🔴	NO ❗	
IERC3156FlashLender	Interface				
└	maxFlashLoan	External ❗		NO ❗	
└	flashFee	External ❗		NO ❗	
└	flashLoan	External ❗	🔴	NO ❗	
PoolLibrary	Library				
└	_checkWeightsValidity	Internal 🔒			
Initializable	Implementation				
└	isConstructor	Private 🔐			
SimpleMosaicOracle	Implementation	Initializable, IMosaicOracle			
└	initialize	External ❗	🔴	initializer	
└	getAdmin	External ❗		NO ❗	
└	setDefaultMiddleHop	External ❗	🔴	NO ❗	msg.sender == admin
└	setMiddleHop	External ❗	🔴	NO ❗	msg.sender == admin
└	batchSetMiddleHop	External ❗	🔴	NO ❗	msg.sender == admin
└	getMiddleHop	External ❗		NO ❗	
└	getPrice	External ❗		NO ❗	
└	getBidPrice	External ❗		NO ❗	
└	_getPrice	Private 🔐			
└	getBatchPrice	External ❗		NO ❗	
└	getPoolPrice	External ❗		NO ❗	
└	_getBatchPrice	Private 🔐			
└	getUsdPrice	External ❗		NO ❗	
└	getUsdBatchPrice	External ❗		NO ❗	

└	Function Name	Visibility	Mutability	Modifiers	Access Control
└	getUsdPoolPrice	External ❗		NO ❗	
└	getUsdPoolPricePerLp	External ❗		NO ❗	
└	_getUsdBatchPrice	Private 🔐			
└	_getDirectPairPrice	Private 🔐			
└	_calculatePrice	Private 🔐			
└	_getPairAddress	Private 🔐			
└	consultPrice	External ❗	🛑	NO ❗	
└	consultBidPrice	External ❗	🛑	NO ❗	
└	_consultPrice	Private 🔐	🛑		
└	consultBatchPrice	External ❗	🛑	NO ❗	
└	consultPoolPrice	External ❗	🛑	NO ❗	
└	_consultBatchPrice	Private 🔐	🛑		
└	consultUsdPrice	External ❗	🛑	NO ❗	
└	consultUsdBatchPrice	External ❗	🛑	NO ❗	
└	consultUsdPoolPrice	External ❗	🛑	NO ❗	
└	consultUsdPoolPricePerLp	External ❗	🛑	NO ❗	
└	_consultUsdBatchPrice	Private 🔐	🛑		
└	_consultDirectPairPrice	Private 🔐	🛑		
└	_consultPairAddress	Private 🔐	🛑		
FixedPoint	Library				
└	mulDown	Internal 🔒			
└	mulUp	Internal 🔒			
└	divDown	Internal 🔒			
└	divUp	Internal 🔒			
└	powDown	Internal 🔒			
└	powUp	Internal 🔒			
└	complement	Internal 🔒			
└	pow	Internal 🔒			
└	exp	Internal 🔒			
└	log	Internal 🔒			
└	ln	Internal 🔒			
└	_ln	Private 🔐			
└	_ln_36	Private 🔐			
Pool	Implementation	IPool			
└	VERSION	External ❗		NO ❗	
└	Live	Internal 🔒			
└	owner	Public ❗		NO ❗	
└	_checkOwner	Internal 🔒			
└	transferOwnership	External ❗	🛑	onlyOwner	onlyOwner
└	_transferOwnership	Internal 🔒	🛑		
└	_checkWeightsValidity	Internal 🔒			
└	initialize	External ❗	🛑	NO ❗	?
└	updateMinDuration	External ❗	🛑	NO ❗	?
└	renouncePool	External ❗	🛑	onlyOwner	onlyOwner
└	initialLiquidityProvided	External ❗	🛑	NO ❗	msg.sender == vaultAddress
└	setManagers	Public ❗	🛑	onlyOwner	onlyOwner
└	balanceOf	External ❗		NO ❗	
└	transfer	External ❗	🛑	NO ❗	
└	safeTransfer	External ❗	🛑	NO ❗	
└	_transfer	Private 🔐	🛑		
└	transferFrom	External ❗	🛑	NO ❗	
└	safeTransferFrom	External ❗	🛑	NO ❗	
└	_transferFrom	Private 🔐	🛑		
└	emitTransfer	External ❗	🛑	NO ❗	msg.sender == vaultAddress
└	approve	External ❗	🛑	NO ❗	
└	allowance	External ❗		NO ❗	
└	calcMintAmounts	Public ❗		NO ❗	
└	_calcMintAmounts	Private 🔐			
└	calcBurnAmounts	Public ❗		NO ❗	
└	_calcBurnAmounts	Private 🔐			
└	_updateTrailingFee	Internal 🔒			
└	_mint	External ❗	🛑	mustBeUnlocked	msg.sender == vaultAddress
└	_burn	External ❗	🛑	NO ❗	msg.sender == vaultAddress
└	_calcOutGivenIn	Public ❗		NO ❗	
└	_calcInGivenOut	Public ❗		NO ❗	
└	getAmountOut	External ❗		NO ❗	
└	getAmountIn	External ❗		NO ❗	
└	queryExactTokensForTokens	External ❗		NO ❗	

└	Function Name	Visibility	Mutability	Modifiers	Access Control
└	queryTokensForExactTokens	External ❗		NO ❗	
└	addToken	External ❗	🔴	onlyPoolOwnerOrManager	onlyPoolOwnerOrManager
└	removeToken	External ❗	🔴	NO ❗	
└	updateBuyFee	External ❗	🔴	onlyOwner	onlyOwner
└	removeExpired	Private 🔐	🔴		
└	checkConflictingAuctions	Private 🔐			
└	dutchAuction	External ❗	🔴	onlyPoolOwnerOrManager	onlyPoolOwnerOrManager
└	stopDutchAuction	External ❗	🔴	onlyPoolOwnerOrManager	onlyPoolOwnerOrManager
└	updateWeights	External ❗	🔴	onlyPoolOwnerOrManager	onlyPoolOwnerOrManager
└	calcWeight	Public ❗		NO ❗	
└	setLocked	External ❗	🔴	onlyOwner	onlyOwner
└	getTokens	Public ❗		NO ❗	
└	getReserves	Public ❗		NO ❗	
└	getWeights	Public ❗		NO ❗	
└	getWeight	Public ❗		NO ❗	
└	totalSupply	External ❗		NO ❗	
└	getTrailingFeeAmount	External ❗		NO ❗	
└	_totalSupply	Private 🔐			
└	_totalSupply	Private 🔐			
└	updatePerfFeePricePerLp	External ❗	🔴	NO ❗	!!!
└	_updatePerfFeePricePerLp	Private 🔐	🔴		
└	_getTrailingFeeAmount	Private 🔐			
└	getPoolFees	External ❗		NO ❗	
PoolFactory	Implementation	Ownable, Governed			
└		Public ❗	🔴	NO ❗	
└	addSystemAddress	External ❗	🔴	onlyOwner	onlyOwner
└	removeSystemAddress	External ❗	🔴	onlyOwner	onlyOwner
└	_selfManageMeBefore	Internal 🔒	🔴		
└	_selfManageMeAfter	Internal 🔒	🔴		
└	_onBeforeEmergencyChange	Internal 🔒	🔴		
└	create	External ❗	🔴	Live	
└	setMinInitialLiquidityValue	External ❗	🔴	Live onlyOwner	onlyOwner
└	setFeeBurner	External ❗	🔴	Live onlyOwner	onlyOwner
└	setCreatePoolFeeAmount	External ❗	🔴	Live onlyOwner	onlyOwner
QueryApi	Implementation	IQueryApi, Ownable			
└		Public ❗	🔴	NO ❗	
└	setVaultaddress	External ❗	🔴	onlyOwner	onlyOwner
└	getPool	Public ❗		NO ❗	
└	getPools	External ❗		NO ❗	
Register	Implementation	IRegister, Controllable, Governed			
└		Public ❗	🔴	NO ❗	
└	_selfManageMeBefore	Internal 🔒	🔴		
└	_selfManageMeAfter	Internal 🔒	🔴		
└	_onBeforeEmergencyChange	Internal 🔒	🔴		
└	approvePools	External ❗	🔴	Live onlyController	onlyController
└	approvePoolsById	External ❗	🔴	Live onlyController	onlyController
└	revokePools	External ❗	🔴	Live onlyController	onlyController
└	revokePoolsById	External ❗	🔴	Live onlyController	onlyController
└	_approvePool	Internal 🔒	🔴		
└	approveTraders	External ❗	🔴	Live onlyController	onlyController
└	revokeTraders	External ❗	🔴	Live onlyController	onlyController
└	_approveTrader	Internal 🔒	🔴		
└	isApprovedPool	External ❗		NO ❗	
└	isApprovedPool	External ❗		NO ❗	
└	isApprovedTrader	External ❗		NO ❗	
└	isWhitelisted	External ❗		NO ❗	
└	isBlacklisted	External ❗		NO ❗	
└	_isBlacklisted	Internal 🔒			
└	addWhitelist	External ❗	🔴	Live onlyController	onlyController
└	addWhitelistBulk	External ❗	🔴	Live onlyController	onlyController
└	addBlacklist	External ❗	🔴	Live onlyController	onlyController
└	removeWhitelist	External ❗	🔴	Live onlyController	onlyController
└	removeBlacklist	Public ❗	🔴	Live onlyController	onlyController
Affiliate	Implementation	IAffiliate, Controllable, Governed			
└		Public ❗	🔴	NO ❗	
└	_selfManageMeBefore	Internal 🔒	🔴		

└	Function Name	Visibility	Mutability	Modifiers	Access Control
└	_selfManageMeAfter	Internal 🔒	🔴		
└	_onBeforeEmergencyChange	Internal 🔒	🔴		
└	getCommissionLevelsForRanks	External ❗		NO ❗	
└	getConfig	External ❗		NO ❗	
└	updateConfig	External ❗	🔴	Live onlyOwner	onlyOwner
└	getUserData	External ❗		NO ❗	
└	getLevelCount	External ❗		NO ❗	
└	getLevelDetails	External ❗		NO ❗	
└	getAllLevelDetails	External ❗		NO ❗	
└	addNextLevel	External ❗	🔴	Live onlyOwner	onlyOwner
└	updateLevel	External ❗	🔴	Live onlyOwner	onlyOwner
└	setUserProfileContract	External ❗	🔴	Live onlyOwner	onlyOwner
└	setUserOracleContract	External ❗	🔴	Live onlyOwner	onlyOwner
└	getLevelsAndConversionAndClaimLimitForRank	External ❗		NO ❗	
└	userStakeChanged	External ❗	🔴	Live onlyController	onlyController
└	registerUserPurchaseAsTokens	External ❗	🔴	Live onlyController	onlyController
└	registerUserPurchase	External ❗	🔴	Live onlyController	onlyController
└	_registerUserPurchase	Internal 🔒	🔴		
└	getAffiliateUserData	External ❗		NO ❗	
└	getUserRank	External ❗		NO ❗	
└	getUserPurchaseAmount	External ❗		NO ❗	
└	getReferralPurchaseAmount	External ❗		NO ❗	
└	getTraderlPurchaseAmount	External ❗		NO ❗	
└	_getUserRank	Internal 🔒			
└	_getRankByKdxStake	Internal 🔒			
└	_getRankByUserPurchase	Internal 🔒			
└	_getRankByReferralPurchase	Internal 🔒			
└	_getRankByReferralCount	Internal 🔒			
└	_getRankByTraderPurchase	Internal 🔒			
AffiliateBooster	Implementation	SimpleEntropy, IAffiliateBooster, Ownable, Governed			
└		Public ❗	🔴	NO ❗	
└	_selfManageMeBefore	Internal 🔒	🔴		
└	_selfManageMeAfter	Internal 🔒	🔴		
└	_onBeforeEmergencyChange	Internal 🔒	🔴		
└	closeEpoch	External ❗	🔴	NO ❗	
└	getCurrentEpoch	External ❗		NO ❗	
└	getCurrentEpochState	External ❗		NO ❗	
└	getCurrentEpochStart	External ❗		NO ❗	
└	getEpochSize	External ❗		NO ❗	
└	getEpochTime	External ❗		NO ❗	
└	getTotalTicketCount	External ❗		NO ❗	
└	getTotalTicketsSpent	External ❗		NO ❗	
└	getEpochWeights	External ❗		NO ❗	
└	getFeeToken	External ❗		NO ❗	
└	getFeeTokenBurnable	External ❗		NO ❗	
└	getTotalPlayerWeight	External ❗		NO ❗	
└	getTicketPrice	External ❗		NO ❗	
└	setController	External ❗	🔴	Live onlyOwner	onlyOwner
└	setCallerAllowed	External ❗	🔴	Live onlyOwner	onlyOwner
└	_setCallerAllowed	Internal 🔒	🔴		
└	setEpochWeights	External ❗	🔴	Live onlyOwner	onlyOwner
└	_setFeeTokenBurnable	Internal 🔒	🔴		
└	setFeeTo	External ❗	🔴	Live onlyOwner	onlyOwner
└	purchaseTicket	External ❗	🔴	Live closeEpochIfTimeUp	
└	purchaseTicket	Public ❗	🔴	Live closeEpochIfTimeUp	
└	allocateTicket	External ❗	🔴	Live closeEpochIfTimeUp	msg.sender == controller
└	_allocateTicket	Internal 🔒	🔴		
└	pickNextReferral	External ❗	🔴	Live closeEpochIfTimeUp	allowedCaller[msg.sender]
└	_incrementEpoch	Internal 🔒	🔴		
└	getState	External ❗		NO ❗	
└	getEpochStates	External ❗		NO ❗	
└	getEpochTickets	External ❗		NO ❗	
└	getEpochTicketsCount	External ❗		NO ❗	
└	getEpochTicket	External ❗		NO ❗	
└	getUserTickets	External ❗		NO ❗	
Affiliate_Storage	Implementation				
└		Public ❗	🔴	NO ❗	

└	Function Name	Visibility	Mutability	Modifiers	Access Control
└	add_referred_user	External ❗	🔴	onlyOracle pause_state	onlyOracle
└	update_ref_by_rank	External ❗	🔴	onlyOracle	onlyOracle
└	update_referee_balance	External ❗	🔴	onlyOracle pause_state	onlyOracle
└	update_referee_count	External ❗	🔴	onlyOracle pause_state	onlyOracle
└	update_referee_stake	External ❗	🔴	onlyOracle pause_state	onlyOracle
└	update_ref_by_balance	External ❗	🔴	onlyOracle pause_state	onlyOracle
└	update_referee_rank	External ❗	🔴	onlyOracle pause_state	onlyOracle
└	change_oracle	External ❗	🔴	onlyOracle	onlyOracle
└	pause_unpause	External ❗	🔴	onlyOracle	onlyOracle
└	read_affiliate	External ❗		NO ❗	
SimpleAffiliateBoosterController	Implementation	IAffiliateBoosterController, Ownable			
└		Public ❗	🔴	NO ❗	
└	setTicketPrice	External ❗	🔴	onlyOwner	onlyOwner
└	setFeeToken	External ❗	🔴	onlyOwner	onlyOwner
└	setFeeTokenTo	External ❗	🔴	onlyOwner	onlyOwner
└	setFeeTokenBurnable	External ❗	🔴	onlyOwner	onlyOwner
└	purchaseTicket	External ❗	🔴	NO ❗	
└	purchaseTicket	External ❗	🔴	NO ❗	
└	_purchaseTicket	Internal 🔒	🔴		
└	getTicketPrice	External ❗	🔴	NO ❗	!!!
└	epochClosed	External ❗	🔴	NO ❗	msg.sender == affiliateBooster
└	ticketAllocated	External ❗	🔴	NO ❗	msg.sender == affiliateBooster
└	referralPicked	External ❗	🔴	NO ❗	
UserProfile	Implementation	IUserProfile, Controllable, Governed			
└		Public ❗	🔴	NO ❗	
└	_selfManageMeBefore	Internal 🔒	🔴		
└	_selfManageMeAfter	Internal 🔒	🔴		
└	_onBeforeEmergencyChange	Internal 🔒	🔴		
└	initializeRootNode	External ❗	🔴	Live onlyOwner	onlyOwner
└	_initializeRootNode	Internal 🔒	🔴		
└	setAffiliateBooster	External ❗	🔴	Live onlyOwner	onlyOwner
└	setFees	External ❗	🔴	Live onlyOwner	onlyOwner
└	setAffiliate	External ❗	🔴	Live onlyOwner	onlyOwner
└	setUserRank	External ❗	🔴	Live	msg.sender == affiliateContract
└	_setDefaultReferral	Internal 🔒	🔴		
└	getParent	External ❗		NO ❗	
└	getParents	External ❗		NO ❗	
└	getParentsAndParentRanks	External ❗		NO ❗	
└	getParentsAndBuyFeeDiscount	External ❗		NO ❗	
└	getReferralCount	External ❗		NO ❗	
└	getActiveReferralCount	External ❗		NO ❗	
└	getAllReferrals	External ❗		NO ❗	
└	getReferrals	External ❗		NO ❗	
└	getDefaultReferral	External ❗		NO ❗	
└	getUser	External ❗		NO ❗	
└	getUserRank	External ❗		NO ❗	
└	getUserCount	External ❗		NO ❗	
└	userExists	External ❗		NO ❗	
└	increaseActiveReferralCount	External ❗	🔴	NO ❗	msg.sender == affiliateContract
└	registerUser	External ❗	🔴	NO ❗	
└	registerUser	External ❗	🔴	onlyController	onlyController
└	registerUserWoBooster	External ❗	🔴	onlyController	onlyController
└	_registerUser	Internal 🔒	🔴		
FlashLoans	Implementation				
└		Public ❗	🔴	NO ❗	
└	flashLoan	External ❗	🔴	NO ❗	
Swaps	Implementation	ISwaps, Governed			
└	_approveToken	Internal 🔒	🔴		
└	addTokens	External ❗	🔴	NO ❗	!!!
└	removeTokens	External ❗	🔴	NO ❗	!!!
└		Public ❗	🔴	NO ❗	
└	_selfManageMeBefore	Internal 🔒	🔴		
└	_selfManageMeAfter	Internal 🔒	🔴		
└	_onBeforeEmergencyChange	Internal 🔒	🔴		
└	getPrice	External ❗		NO ❗	
└	quickQuoteMint	External ❗		NO ❗	

└	Function Name	Visibility	Mutability	Modifiers	Access Control
└	quickQuoteBurn	External ❗		NO ❗	
└	quoteMint	External ❗		NO ❗	
└	quoteBurn	External ❗		NO ❗	
└	mint	External ❗	🔴	Live	
└	burn	External ❗	🔴	Live	
└	quickMint	External ❗	🔴	Live	
└	quickBurn	External ❗	🔴	Live	
└	swapQuote	External ❗		NO ❗	
Vault	Implementation	IVault, IERC3156FlashLender, ReentrancyGuard, Governed			
└	getVaultState	External ❗		NO ❗	
└	getAuctionCount	External ❗		NO ❗	
└	getAuction	External ❗		NO ❗	
└	getPoolState	External ❗		NO ❗	
└		Public ❗	🔴	NO ❗	
└	setEmergencyMode	External ❗	🔴	NO ❗	onlyManagersFun
└	_setEmergencyMode	Internal 🔒	🔴		
└	setPoolEmergencyMode	External ❗	🔴	NO ❗	!isManagerFun(msg.sender) && msg.sender != poolAddress
└	LiveFun	Internal 🔒			
└	selfManageMe	External ❗	🔴	NO ❗	onlyManagersFun
└	_selfManageMeBefore	Internal 🔒	🔴		
└	_selfManageMeAfter	Internal 🔒	🔴		
└	_onBeforeEmergencyChange	Internal 🔒	🔴		
└	_registerUser	Internal 🔒	🔴		
└	_registerUserPurchase	Internal 🔒	🔴		
└	onlyAdmin	Internal 🔒			
└	isAdmin	Public ❗		NO ❗	
└	AddRemoveAdmin	External ❗	🔴	NO ❗	onlyManagersFun
└	AddRemoveBoostedPool	External ❗	🔴	NO ❗	onlyAdmin
└	AddRemoveZeroFeeAddress	External ❗	🔴	NO ❗	onlyAdmin
└	setWhitelistedTokens	External ❗	🔴	NO ❗	onlyAdmin
└	isTokenWhitelisted	Public ❗		NO ❗	
└	getInternalBalance	Public ❗		NO ❗	
└	getInternalBalances	Public ❗		NO ❗	
└	depositToInternalBalance	External ❗	🔴	NO ❗	
└	depositToInternalBalanceToAddress	External ❗	🔴	NO ❗	
└	withdrawFromInternalBalance	External ❗	🔴	NO ❗	
└	withdrawFromInternalBalanceToAddress	External ❗	🔴	NO ❗	
└	transferInternalBalance	External ❗	🔴	NO ❗	
└	_depositToInternalBalance	Internal 🔒	🔴		
└	_withdrawFromInternalBalance	Internal 🔒	🔴		
└	_transferInternalBalance	Internal 🔒	🔴		
└	_transferInternalBalance	Internal 🔒	🔴		
└	_mintToInternalBalance	Internal 🔒	🔴		
└	_burnFromInternalBalance	Internal 🔒	🔴		
└	_calculateTrailingAndPerformanceFee	Internal 🔒	🔴		
└	allocateTrailingAndPerformanceFee	External ❗	🔴	NO ❗	
└	_onBeforeBalanceTransfer	Internal 🔒	🔴		
└	transferFromAsTokenContract	External ❗	🔴	NO ❗	_poolAddressToId[msg.sender] == 0
└	mintAsTokenContract	External ❗	🔴	NO ❗	_poolAddressToId[msg.sender] == 0
└	burnAsTokenContract	External ❗	🔴	NO ❗	_poolAddressToId[msg.sender] == 0
└	disableFees	External ❗	🔴	NO ❗	?
└	poolIdToAddress	External ❗		NO ❗	
└	poolAddressToId	External ❗		NO ❗	
└	userJoinedPools	External ❗		NO ❗	
└	userOwnedPools	External ❗		NO ❗	
└	getPoolTokens	External ❗		NO ❗	
└	getPoolTokensByAddr	External ❗		NO ❗	
└	_getPoolTokensByAddr	Internal 🔒			
└	getPoolTotalSupplyBaseByAddr	External ❗		NO ❗	
└	_consultPoolPricePerLpByAddr	Internal 🔒	🔴		
└	getPoolValueByAddr	External ❗		NO ❗	
└	getPoolTotalSupplyBase	External ❗		NO ❗	
└	registerPool	External ❗	🔴	onlyFactory	onlyFactory
└	registerTokens	External ❗	🔴	NO ❗	_poolId != 0
└	addInitialLiquidity	External ❗	🔴	nonReentrant	msg.sender != poolFactory && msg.sender != _poolAddress && msg.sender != Ipool(_poolAddress).owner()

└	Function Name	Visibility	Mutability	Modifiers	Access Control
└	deregisterToken	External ❗	🔴	NO ❗	_poolId != 0
└	Mint	External ❗	🔴	nonReentrant	
└	Burn	External ❗	🔴	nonReentrant	?
└	swap	Public ❗	🔴	nonReentrant	
└	multiSwap	External ❗	🔴	nonReentrant	
└	getAuctionInfo	External ❗		NO ❗	
└	isRunning	External ❗		NO ❗	
└	startAuction	External ❗	🔴	nonReentrant	poolId != 0
└	stopAuction	External ❗	🔴	nonReentrant	msg.sender != auctions[auctionId].poolAddress
└	getBidPriceAt	Public ❗		NO ❗	
└	getBidPrice	Public ❗		NO ❗	
└	_getBidPrice	Internal 🔒			
└	bid	External ❗	🔴	nonReentrant	
└	_handlerForPoolOfUser	Internal 🔒	🔴		
└	flashLoan	External ❗	🔴	nonReentrant	
└	flashFee	External ❗		NO ❗	
└	_flashFee	Internal 🔒			
└	maxFlashLoan	External ❗		NO ❗	
└	emergencyWithdraw	External ❗	🔴	NO ❗	!poolStates[poolAddress].poolEmergencyMode && !vaultState.emergencyMode
└	emergencyWithdrawAll	External ❗	🔴	NO ❗	onlyAdmin
└	emergencyWithdrawPool	External ❗	🔴	NO ❗	onlyAdmin
Arrays	Library				
└	findUpperBound	Internal 🔒			
Controllable	Implementation	Ownable			
└	addController	External ❗	🔴	onlyOwner	
└	_addController	Internal 🔒	🔴		
└	removeController	External ❗	🔴	onlyOwner	
└	_removeController	Internal 🔒	🔴		
└	isController	Public ❗		NO ❗	
└	_isController	Internal 🔒			
IERC20Burnable	Interface	IERC20			
└	burn	External ❗	🔴	NO ❗	
Ownable	Implementation				
└		Public ❗	🔴	NO ❗	
└	owner	Public ❗		NO ❗	
└	_checkOwner	Internal 🔒			
└	renounceOwnership	Public ❗	🔴	onlyOwner	onlyOwner
└	transferOwnership	Public ❗	🔴	onlyOwner	onlyOwner
└	_transferOwnership	Internal 🔒	🔴		
SimpleEntropy	Implementation	ISimpleEntropy			
└	simpleRandom	External ❗		NO ❗	
└	simpleRandom	External ❗		NO ❗	
└	_simpleRandom	Internal 🔒			
SystemRole	Implementation	AccessControl, Ownable, ISystemRole			
└		Public ❗	🔴	NO ❗	
└	transferOwnership	Public ❗	🔴	onlyOwner	onlyOwner
└	addSystemAdmin	Public ❗	🔴	onlyAdmin	onlyAdmin
└	removeSystemAdmin	Public ❗	🔴	onlyAdmin	onlyAdmin
└	addTrader	Public ❗	🔴	onlyAdmin	onlyAdmin
└	removeTrader	Public ❗	🔴	onlyAdmin	onlyAdmin
└	addInvestor	Public ❗	🔴	onlyAdmin	onlyAdmin
└	removeInvestor	Public ❗	🔴	onlyAdmin	onlyAdmin
└	addDepositLocker	Public ❗	🔴	onlyAdmin	onlyAdmin
└	removeDepositLocker	Public ❗	🔴	onlyAdmin	onlyAdmin
└	addAdmin	Public ❗	🔴	onlyAdmin	onlyAdmin
└	removeAdmin	Public ❗	🔴	onlyAdmin	onlyAdmin
└	_isSystemAdmin	Internal 🔒			
└	_isTrader	Internal 🔒			
└	_isInvestor	Internal 🔒			
└	_isAdmin	Internal 🔒			
└	_isDepositLocker	Internal 🔒			
└	checkSystemAdmin	External ❗		NO ❗	
└	checkTrader	External ❗		NO ❗	
└	checkInvestor	External ❗		NO ❗	
└	checkDepositLocker	External ❗		NO ❗	

└	Function Name	Visibility	Mutability	Modifiers	Access Control
└	checkAdmin	External ❗		NO ❗	
SystemWhitelist	Implementation	SystemRole, ISystemWhitelist			
└		Public ❗	🔴	NO ❗	
└	transferOwnership	Public ❗	🔴	onlyOwner	onlyOwner
└	isWhitelisted	External ❗		onlySystemAdmin	onlySysemAdmin
└	isBlacklisted	External ❗		onlySystemAdmin	onlySysemAdmin
└	_isBlacklisted	Internal 🔒			
└	addWhitelist	External ❗	🔴	onlySystemAdmin	onlySysemAdmin
└	addWhitelistBulk	External ❗	🔴	onlySystemAdmin	onlySysemAdmin
└	addBlacklist	External ❗	🔴	onlySystemAdmin	onlySysemAdmin
└	removeWhitelist	External ❗	🔴	onlySystemAdmin	onlySysemAdmin
└	removeBlacklist	Public ❗	🔴	onlySystemAdmin	onlySysemAdmin
PancakeERC20	Implementation	IPancakeERC20			
└		Public ❗	🔴	NO ❗	
└	_mint	Internal 🔒	🔴		
└	_burn	Internal 🔒	🔴		
└	_approve	Private 🔐	🔴		
└	_transfer	Private 🔐	🔴		
└	approve	External ❗	🔴	NO ❗	
└	transfer	External ❗	🔴	NO ❗	
└	transferFrom	External ❗	🔴	NO ❗	
└	permit	External ❗	🔴	NO ❗	
PancakeFactory	Implementation	IPancakeFactory			
└		Public ❗	🔴	NO ❗	
└	allPairsLength	External ❗		NO ❗	
└	createPair	External ❗	🔴	NO ❗	
└	setFeeTo	External ❗	🔴	NO ❗	
└	setFeeToSetter	External ❗	🔴	NO ❗	
PancakePair	Implementation	IPancakePair, PancakeERC20			
└	getReserves	Public ❗		NO ❗	
└	_safeTransfer	Private 🔐	🔴		
└		Public ❗	🔴	NO ❗	
└	initialize	External ❗	🔴	NO ❗	
└	_update	Private 🔐	🔴		
└	_mintFee	Private 🔐	🔴		
└	mint	External ❗	🔴	lock	
└	burn	External ❗	🔴	lock	
└	swap	External ❗	🔴	lock	
└	skim	External ❗	🔴	lock	
└	sync	External ❗	🔴	lock	
PancakeRouter	Implementation	IPancakeRouter02			
└		Public ❗	🔴	NO ❗	
└		External ❗	💵	NO ❗	
└	_addLiquidity	Internal 🔒	🔴		
└	addLiquidity	External ❗	🔴	ensure	
└	addLiquidityETH	External ❗	💵	ensure	
└	removeLiquidity	Public ❗	🔴	ensure	
└	removeLiquidityETH	Public ❗	🔴	ensure	
└	removeLiquidityWithPermit	External ❗	🔴	NO ❗	
└	removeLiquidityETHWithPermit	External ❗	🔴	NO ❗	
└	removeLiquidityETHSupportingFeeOnTransferTokens	Public ❗	🔴	ensure	
└	removeLiquidityETHWithPermitSupportingFeeOnTransferTokens	External ❗	🔴	NO ❗	
└	_swap	Internal 🔒	🔴		
└	swapExactTokensForTokens	External ❗	🔴	ensure	
└	swapTokensForExactTokens	External ❗	🔴	ensure	
└	swapExactETHForTokens	External ❗	💵	ensure	
└	swapTokensForExactETH	External ❗	🔴	ensure	
└	swapExactTokensForETH	External ❗	🔴	ensure	
└	swapETHForExactTokens	External ❗	💵	ensure	
└	_swapSupportingFeeOnTransferTokens	Internal 🔒	🔴		
└	swapExactTokensForTokensSupportingFeeOnTransferTokens	External ❗	🔴	ensure	
└	swapExactETHForTokensSupportingFeeOnTransferTokens	External ❗	💵	ensure	
└	swapExactTokensForETHSupportingFeeOnTransferTokens	External ❗	🔴	ensure	
└	quote	Public ❗		NO ❗	
└	getAmountOut	Public ❗		NO ❗	
└	getAmountIn	Public ❗		NO ❗	

└	Function Name	Visibility	Mutability	Modifiers	Access Control
└	getAmountsOut	Public ❗		NO ❗	
└	getAmountsIn	Public ❗		NO ❗	
IERC20Pan	Interface				
└	name	External ❗		NO ❗	
└	symbol	External ❗		NO ❗	
└	decimals	External ❗		NO ❗	
└	totalSupply	External ❗		NO ❗	
└	balanceOf	External ❗		NO ❗	
└	allowance	External ❗		NO ❗	
└	approve	External ❗	🛑	NO ❗	
└	transfer	External ❗	🛑	NO ❗	
└	transferFrom	External ❗	🛑	NO ❗	
IWETH	Interface				
└	deposit	External ❗	💵	NO ❗	
└	transfer	External ❗	🛑	NO ❗	
└	withdraw	External ❗	🛑	NO ❗	
Babylonian	Library				
└	sqrt	Internal 🔒			
BitMath	Library				
└	mostSignificantBit	Internal 🔒			
└	leastSignificantBit	Internal 🔒			
FixedPoint	Library				
└	encode	Internal 🔒			
└	encode144	Internal 🔒			
└	decode	Internal 🔒			
└	decode144	Internal 🔒			
└	mul	Internal 🔒			
└	muli	Internal 🔒			
└	muluq	Internal 🔒			
└	divuq	Internal 🔒			
└	fraction	Internal 🔒			
└	reciprocal	Internal 🔒			
└	sqrt	Internal 🔒			
FullMath	Library				
└	fullMul	Internal 🔒			
└	fullDiv	Private 🔐			
└	mulDiv	Internal 🔒			
Math	Library				
└	min	Internal 🔒			
└	sqrt	Internal 🔒			
PancakeLibrary	Library				
└	sortTokens	Internal 🔒			
└	pairFor	Internal 🔒			
└	getReserves	Internal 🔒			
└	quote	Internal 🔒			
└	getAmountOut	Internal 🔒			
└	getAmountIn	Internal 🔒			
└	getAmountsOut	Internal 🔒			
└	getAmountsIn	Internal 🔒			
PancakeOracleLibrary	Library				
└	currentBlockTimestamp	Internal 🔒			
└	currentCumulativePrices	Internal 🔒			
SafeMath	Library				
└	add	Internal 🔒			
└	sub	Internal 🔒			
└	mul	Internal 🔒			
TransferHelper	Library				
└	safeApprove	Internal 🔒	🛑		
└	safeTransfer	Internal 🔒	🛑		
└	safeTransferFrom	Internal 🔒	🛑		
└	safeTransferETH	Internal 🔒	🛑		
UQ112x112	Library				
└	encode	Internal 🔒			
└	uqdiv	Internal 🔒			

Glossary

Mosaic Alpha Smart Contract Analysis Appendix D

Contents

Glossary

Mosaic-specific terminology

Token basket/pool

Mosaic Baskets serve as decentralized alternatives to ETFs designed for the crypto sector and actively managed funds. These smart-contract-enabled crypto products represent baskets of crypto assets, and the system guarantees their tracking of the exchange rates by actually buying the components of the represented asset basket upon the opening of the position and selling them upon closing the position.

Rebalancing pool

In the rebalancing model, the distribution of the components is determined by the Basket Manager based on the **percent distribution of the assets' value** in stablecoin. In the case of the Rebalancing Baskets, the percentage value-weighted distribution is constant, but the actual nominal per-unit crypto asset components change in relation to the exchange rates.

Standard pool

In the case of a Standard model, the components and the way the chosen assets are distributed can be set by the Basket Manager as **exact nominal amount**s, so the relative nominal composition of the basket remains a fixed number of units, so the value of the Basket per unit is determined. With Standard Baskets, the distribution of the contents is independent of the market's direction and when the exchange rates for the individual components of the contents change, the distribution doesn't.

Kodexa (KDX)

The utility token of the Mosaic platform. Kodexa (KDX) holders can earn more benefits by staking their KDX tokens. There is a fix amount of KDX in circulation and KDX can be swapped on the Mosaic swap.

Mosaic swap

Mosaic swap is the platform's marketplace. The service uses the liquidity pools as backends. The exchange rates on the swap are adjusted to help achieve the percantage goals of the rebalancing baskets through attracting arbitrageurs.

Fees

Trailing fee (operation, maintenance, and development fee)

This fee is being continuously deducted to contribute to the costs of operation.

Buy fee (initial cost)

An initial fee that is charged immediately after investing.

Performance fee (profit sharing cost)

This fee is deducted only in the case of an increase in the value of the baskets. The platform monitors the basket value continuously and the performance fee is applicable when the current basket value is higher than at any time before. The performance fee is counted from the difference between the previous highest value and the current value.

Affiliate system

Mosaic Alpha operates an affiliate system similar to other financial market participants. The basis of the affiliate system and computing commission

- There are two levels of depth you can get a commission.
- You can get a commission from users that used your affiliate link and invested some amount as well.
- You can only get commission after staking the required amount of KDX
- The more your affiliates are, the more they invest, and the more you stake the higher your rank is and your commission is closer to the maximum reachable.

Affiliate booster

Helps in selecting a referrer for new users who registered without an affiliate link. KDX can be exchanged for tickets and the referrer of the new user will be randomly selected from the ticket holders.

Liquidity pool (LP) token

The baskets (pools) can be traded as liquidity pool ERC20 tokens.

Discount level

In order to gain access to advantages on the Mosaic platform, users can level up by fulfilling various criteria.

Kodexa staking

Staking KDX allows earning some rewards, which can be a lesser paid fee, faster upgrading in the affiliate system, and getting more commission.

General fintech terminology

Swap

Swap refers to exchanging one crypto asset for another, like swapping your BTC into ETH.

DEX

A decentralized exchange (better known as a DEX) is a peer-to-peer marketplace where transactions occur directly between crypto traders. See https://www.coinbase.com/learn/crypto-basics/what-is-a-dex

USDT

Tether (USDT) is a stablecoin that is pegged to the U.S. dollar. The peg to a traditional currency, often backed by collateral reserves made up entirely or mostly of the pegged currency, ensures stablecoins aren't subject to the same price volatility as more speculative cryptocurrencies like Bitcoin.

Staking

Staking is when you lock crypto assets for a set period of time to help support the operation of a blockchain. See https://www.forbes.com/advisor/in/investing/cryptocurrency/what-is-staking-in-crypto/

ERC20 token

ERC-20 is a technical standard used to issue and implement tokens on the Ethereum blockchain.

Flash loan

A flash loan is a type of uncollateralized loan that lets a user borrow assets with no upfront collateral as long as the borrowed assets are paid back within the same blockchain transaction.

(In the case of collateralized lending, borrowers need to put up capital (collateral) to borrow funds. If the borrower fails to meet the terms of the loan, the lender can still cover the loan using the borrower’s collateral. Flash loans don’t have this requirement; the loan can only exist if the borrower pays it back within the same transaction. As a result, defaulting on a flash loan is not possible, since the entire transaction would simply revert)

Token burning

Token burning means removing coins from the overall supply of a cryptocurrency. This typically involves sending the coins or tokens to a wallet with no known private keys. This wallet can only receive assets, thus effectively making them inaccessible.

Minting

I think the use of this term is a bit confusing here, because based on what I was able to find, minting usually refers to the act of "generating new coins by authenticating data, creating new blocks, and recording the information onto the blockchain through a “proof of stake” protocol", whereas in this context it only refers to the creation of new LP tokens to achieve the effect of deducting fees through inflation.

Utility token

A digital token of a blockchain application that is issued in order to fund development of the application and that can be later used to purchase a good or service offered by application

Contract Exploration

Mosaic Alpha Smart Contract Analysis Appendix E

Contents

Home

This document contains the individual documentation of the contracts.

Affiliate.sol

Responsibilities

Responsible for affiliate logic. Keeps track of the discount levels (which depend on the amount of KDX a user stakes and the amount of pools they purchased in their lifetime).

Interactions with other contracts

Uses the `IMosaicOracle` interface to query the USDT prices of tokens from the oracle.

- IMosaicOracle::consultUsdBatchPrice

Uses the `IUserProfile` interface to register users, get and set their ranks, get their referrers and increase their active referral count.

- IUserProfile::registerUser
- IUserProfile::getParent
- IUserProfile::getUserRank
- IUserProfile::setUserRank
- IUserProfile::increaseActiveReferralCount

Issues

Potential reentrancy in `_registerUserPurchase`. False positive.

```
Reentrancy in Affiliate._registerUserPurchase(address,address,address,uint256) (contracts/mosaic-alpha-
contracts/User/Affiliate.sol#220-279):
    External calls:
    - IUserProfile(userProfile).registerUser(user,referredBy) (contracts/mosaic-alpha-contracts/User/Affiliate.sol#221)
    - IUserProfile(userProfile).registerUser(trader,address(0)) (contracts/mosaic-alpha-contracts/User/Affiliate.sol#223)
    - IUserProfile(userProfile).increaseActiveReferralCount(referredBy) (contracts/mosaic-alpha-contracts/User/Affiliate.sol#232)
    - IUserProfile(userProfile).setUserRank(referredBy,newRank) (contracts/mosaic-alpha-contracts/User/Affiliate.sol#247)
    State variables written after the call(s):
    - userData[referredBy].affiliateRevision = revision (contracts/mosaic-alpha-contracts/User/Affiliate.sol#248)
Reentrancy in Affiliate._registerUserPurchase(address,address,address,uint256) (contracts/mosaic-alpha-
contracts/User/Affiliate.sol#220-279):
    External calls:
    - IUserProfile(userProfile).registerUser(user,referredBy) (contracts/mosaic-alpha-contracts/User/Affiliate.sol#221)
    - IUserProfile(userProfile).registerUser(trader,address(0)) (contracts/mosaic-alpha-contracts/User/Affiliate.sol#223)
    - IUserProfile(userProfile).increaseActiveReferralCount(referredBy) (contracts/mosaic-alpha-contracts/User/Affiliate.sol#232)
    - IUserProfile(userProfile).setUserRank(referredBy,newRank) (contracts/mosaic-alpha-contracts/User/Affiliate.sol#243)
    - IUserProfile(userProfile).setUserRank(referredBy,newRank) (contracts/mosaic-alpha-contracts/User/Affiliate.sol#247)
    - IUserProfile(userProfile).setUserRank(user,newRank) (contracts/mosaic-alpha-contracts/User/Affiliate.sol#260)
    State variables written after the call(s):
    - userData[user].affiliateRevision = revision (contracts/mosaic-alpha-contracts/User/Affiliate.sol#261)
Reentrancy in Affiliate._registerUserPurchase(address,address,address,uint256) (contracts/mosaic-alpha-
contracts/User/Affiliate.sol#220-279):
    External calls:
    - IUserProfile(userProfile).registerUser(user,referredBy) (contracts/mosaic-alpha-contracts/User/Affiliate.sol#221)
    - IUserProfile(userProfile).registerUser(trader,address(0)) (contracts/mosaic-alpha-contracts/User/Affiliate.sol#223)
    - IUserProfile(userProfile).increaseActiveReferralCount(referredBy) (contracts/mosaic-alpha-contracts/User/Affiliate.sol#232)
    - IUserProfile(userProfile).setUserRank(referredBy,newRank) (contracts/mosaic-alpha-contracts/User/Affiliate.sol#243)
    - IUserProfile(userProfile).setUserRank(referredBy,newRank) (contracts/mosaic-alpha-contracts/User/Affiliate.sol#247)
    - IUserProfile(userProfile).setUserRank(user,newRank) (contracts/mosaic-alpha-contracts/User/Affiliate.sol#256)
    - IUserProfile(userProfile).setUserRank(user,newRank) (contracts/mosaic-alpha-contracts/User/Affiliate.sol#260)
    State variables written after the call(s):
    - userData[trader].traderPurchase += usdAmount (contracts/mosaic-alpha-contracts/User/Affiliate.sol#265)
Reentrancy in Affiliate._registerUserPurchase(address,address,address,uint256) (contracts/mosaic-alpha-
contracts/User/Affiliate.sol#220-279):
    External calls:
    - IUserProfile(userProfile).registerUser(user,referredBy) (contracts/mosaic-alpha-contracts/User/Affiliate.sol#221)
    - IUserProfile(userProfile).registerUser(trader,address(0)) (contracts/mosaic-alpha-contracts/User/Affiliate.sol#223)
    - IUserProfile(userProfile).increaseActiveReferralCount(referredBy) (contracts/mosaic-alpha-contracts/User/Affiliate.sol#232)
    - IUserProfile(userProfile).setUserRank(referredBy,newRank) (contracts/mosaic-alpha-contracts/User/Affiliate.sol#243)
    - IUserProfile(userProfile).setUserRank(referredBy,newRank) (contracts/mosaic-alpha-contracts/User/Affiliate.sol#247)
    - IUserProfile(userProfile).setUserRank(user,newRank) (contracts/mosaic-alpha-contracts/User/Affiliate.sol#256)
    - IUserProfile(userProfile).setUserRank(user,newRank) (contracts/mosaic-alpha-contracts/User/Affiliate.sol#260)
    - IUserProfile(userProfile).setUserRank(trader,newRank) (contracts/mosaic-alpha-contracts/User/Affiliate.sol#275)
    State variables written after the call(s):
    - userData[trader].affiliateRevision = revision (contracts/mosaic-alpha-contracts/User/Aff
```

In:

- `Affiliate.registerUserPurchase` -> `onlyController`
- `Affiliate.registerUserPurchaseAsTokens` -> `onlyController`

Out:

- ✓ `UserProfile.increaseActiveReferralCount`
- ✓ `UserProfile.getParent`
- ✓ `UserProfile.getUserRank`
- -> `UserProfile.registerUser`
    - -> `AffiliateBooster.pickNextReferral`

            - ✓ AffiliateBoosterController.epochClosed
            - ✓ AffiliateBoosterController.referralPicked
        - ✓ Affiliate.getLevelDetails
- -> UserProfile.setUserRank
    - -> Fees.userRankChanged
        - ✓ Affiliate.getLevelsAndConversionAndClaimLimitForRank
    - -> AffiliateBooster.pickNextReferral
        - ✓ AffiliateBoosterController.epochClosed
        - ✓ AffiliateBoosterController.referralPicked
    - ✓ Affiliate.getLevelDetails

Potential reentrancy in `userStakeChanged`. False positive.

```
Reentrancy in Affiliate.userStakeChanged(address,address,uint256) (contracts/mosaic-alpha-contracts/User/Affiliate.sol#163-191):
    External calls:
    - IUserProfile(userProfile).registerUser(user,referredBy) (contracts/mosaic-alpha-contracts/User/Affiliate.sol#164)
    - IUserProfile(userProfile).setUserRank(user,newRank_scope_0) (contracts/mosaic-alpha-contracts/User/Affiliate.sol#188)
    State variables written after the call(s):
    - userData[user].affiliateRevision = revision (contracts/mosaic-alpha-contracts/User/Affiliate.sol#189)
```

In:

- `Affiliate.userStakeChanged` -> `onlyController`

Out:

- ✓ UserProfile.getParent
- ✓ UserProfile.getUserRank
- -> UserProfile.registerUser
    - -> AffiliateBooster.pickNextReferral
        - ✓ AffiliateBoosterController.epochClosed
        - ✓ AffiliateBoosterController.referralPicked
    - ✓ Affiliate.getLevelDetails
- -> UserProfile.setUserRank
    - -> Fees.userRankChanged
        - ✓ Affiliate.getLevelsAndConversionAndClaimLimitForRank
    - -> AffiliateBooster.pickNextReferral
        - ✓ AffiliateBoosterController.epochClosed
        - ✓ AffiliateBoosterController.referralPicked
    - ✓ Affiliate.getLevelDetails

Affiliate_old.sol

Responsibilities

Isn't used, should be ignored.

Interactions with other contracts

Issues

AffiliateBooster.sol

Responsibilities

Responsible for assigning referrers to users who arrived without an affiliate link. Tickets can be bought in exchange for KDX and each time a new user comes without an affiliate link, a referrer is randomly selected from the ticket holders.

Interactions with other contracts

Uses the `IUserProfile` interface to register users

- IUserProfile::registerUser

Uses the `IAffiliateBoosterController` interface to send callbacks to the controller (epoch closed, ticket allocated, referral picked)

- IAffiliateBoosterController::epochClosed
- IAffiliateBoosterController::ticketAllocated
- IAffiliateBoosterController::referralPicked

Uses the `IERC20Burnable` interface to transfer the ticket price to itself (in the fee token) and burn it (or redirect to a given address if set).

- IERC20Burnable::transferFrom
- IERC20Burnable::burn
- IERC20Burnable::transfer

Issues

High

In line 328, `transfer` might return `false`. If it happens, tokens remain with `AffiliateBooster`.

Medium

Potential reentrancy in `purchaseTicket`. False positive.

```
Reentrancy in AffiliateBooster.purchaseTicket(uint256) (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#313-315):
    External calls:
    - purchaseTicket(address(0),_c) (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#314)
        - userProfile.registerUser(_a,_referredBy) (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#353)
        - require(bool,string)(IERC20Burnable(state.feeToken).transferFrom(msg.sender,address(this),_c *
state.ticketPrice),TransferFrom failed) (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#324)
        - IERC20Burnable(state.feeToken).burn(_c * state.ticketPrice) (contracts/mosaic-alpha-
contracts/User/AffiliateBooster.sol#326)
        - IERC20Burnable(state.feeToken).transfer(state.feeTo,_c * state.ticketPrice) (contracts/mosaic-alpha-
contracts/User/AffiliateBooster.sol#328)
        - controller.epochClosed(state.currentEpoch - 1) (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#427)
        - controller.ticketAllocated(_a,_c) (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#362)
    - closeEpochIfTimeUp() (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#313)
        - controller.epochClosed(state.currentEpoch - 1) (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#427)
    State variables written after the call(s):
    - purchaseTicket(address(0),_c) (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#314)
        - epochTickets[state.currentEpoch].push(_a) (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#355)
    - purchaseTicket(address(0),_c) (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#314)
        - epochs[state.currentEpoch].epochStart = uint64(block.timestamp) (contracts/mosaic-alpha-
contracts/User/AffiliateBooster.sol#421)
        - epochs[state.currentEpoch].ticketsBought += uint64(_c) (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#358)
    - purchaseTicket(address(0),_c) (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#314)
        - state.totalPlayerWeight = 0 (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#416)
        - state.currentEpoch ++ (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#418)
        - state.currentEpochStart = uint64(block.timestamp) (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#420)
        - i < state.epochWeights.length && i <= state.currentEpoch (contracts/mosaic-alpha-
contracts/User/AffiliateBooster.sol#422)
        - state.totalPlayerWeight += state.epochWeights[i] * epochTickets[state.currentEpoch - i].length (contracts/mosaic-alpha-
contracts/User/AffiliateBooster.sol#423)
        - state.totalPlayerWeight += state.epochWeights[0] (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#356)
        - state.totalTicketCount += uint64(_c) (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#360)
Reentrancy in AffiliateBooster.purchaseTicket(address,uint256) (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#322-
331):
    External calls:
    - require(bool,string)(IERC20Burnable(state.feeToken).transferFrom(msg.sender,address(this),_c *
state.ticketPrice),TransferFrom failed) (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#324)
    - IERC20Burnable(state.feeToken).burn(_c * state.ticketPrice) (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#326)
    - IERC20Burnable(state.feeToken).transfer(state.feeTo,_c * state.ticketPrice) (contracts/mosaic-alpha-
contracts/User/AffiliateBooster.sol#328)
    - _allocateTicket(msg.sender,_referredBy,_c,state.ticketPrice) (contracts/mosaic-alpha-
contracts/User/AffiliateBooster.sol#330)
        - userProfile.registerUser(_a,_referredBy) (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#353)
        - controller.ticketAllocated(_a,_c) (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#362)
    - closeEpochIfTimeUp() (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#322)
```

```
        - controller.epochClosed(state.currentEpoch - 1) (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#427)
    State variables written after the call(s):
    - _allocateTicket(msg.sender,_referredBy,_c,state.ticketPrice) (contracts/mosaic-alpha-
contracts/User/AffiliateBooster.sol#330)
        - epochTickets[state.currentEpoch].push(_a) (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#355)
    - _allocateTicket(msg.sender,_referredBy,_c,state.ticketPrice) (contracts/mosaic-alpha-
contracts/User/AffiliateBooster.sol#330)
        - epochs[state.currentEpoch].ticketsBought += uint64(_c) (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#358)
    - _allocateTicket(msg.sender,_referredBy,_c,state.ticketPrice) (contracts/mosaic-alpha-
contracts/User/AffiliateBooster.sol#330)
        - state.totalPlayerWeight += state.epochWeights[0] (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#356)
        - state.totalTicketCount += uint64(_c) (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#360)
```

In:

- `AffiliateBooster.purchaseTicket` ->

Out:

- -> `IERC20Burnable.transfer`: Csak fee token-t.
- -> `IERC20Burnable.transferFrom`: Csak fee token-t.
- -> `IERC20Burnable.burn`: Csak fee token-t.
- -> `UserProfile.registerUser`
    - -> `AffiliateBooster.pickNextReferral`
        - ✓ `AffiliateBoosterController.epochClosed`
        - ✓ `AffiliateBoosterController.referralPicked`
    - ✓ `Affiliate.getLevelDetails`
- ✓ `AffiliateBoosterController.ticketAllocated`

Potential reentrancy in `allocateTicket`. False positive.

```
Reentrancy in AffiliateBooster.allocateTicket(address,address,uint256,uint256) (contracts/mosaic-alpha-
contracts/User/AffiliateBooster.sol#340-343):
    External calls:
    - _allocateTicket(_a,_referredBy,_c,_price) (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#342)
        - userProfile.registerUser(_a,_referredBy) (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#353)
        - controller.ticketAllocated(_a,_c) (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#362)
    - closeEpochIfTimeUp() (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#340)
        - controller.epochClosed(state.currentEpoch - 1) (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#427)
    State variables written after the call(s):
    - _allocateTicket(_a,_referredBy,_c,_price) (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#342)
        - epochTickets[state.currentEpoch].push(_a) (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#355)
    - _allocateTicket(_a,_referredBy,_c,_price) (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#342)
        - epochs[state.currentEpoch].ticketsBought += uint64(_c) (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#358)
    - _allocateTicket(_a,_referredBy,_c,_price) (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#342)
        - state.totalPlayerWeight += state.epochWeights[0] (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#356)
        - state.totalTicketCount += uint64(_c) (contracts/mosaic-alpha-contracts/User/AffiliateBooster.sol#360)
```

In:

- `AffiliateBooster.allocateTicket` ->

Out:

- -> `UserProfile.registerUser`
    - -> `AffiliateBooster.pickNextReferral`
        - ✓ `AffiliateBoosterController.epochClosed`
        - ✓ `AffiliateBoosterController.referralPicked`
    - ✓ `Affiliate.getLevelDetails`
- ✓ `AffiliateBoosterController.ticketAllocated`

Arrays.sol

Responsibilities

A helper contract for arrays.

Interactions with other contracts

Does not interact with other contracts.

Issues

Controllable.sol

Responsibilities

A helper class for access control. Admins can add and remove `controller` addresses, which are allowed to mint/burn tokens.

Interactions with other contracts

Does not call other contracts.

Issues

FeeBurner.sol

Responsibilities

KDX which is collected for the AffiliateBooster and other purposes is transferred here and either burned or redirected to a given address based on some policy.

Interactions with other contracts

Uses the `IERC20Burnable` interface to burn tokens

- IERC20Burnable::burn

Uses the `IERC20` interface to get the balance of tokens and to transfer not burnable tokens to a given address

- IERC20::balanceOf
- IERC20::transfer

Issues

Unchecked transfer in line 74. Function `transfer` might return `false`, so no token is transferred to `feeTo`. Depends on the specific implementation of `feeToken`.

Fees.sol

Responsibilities

This contract is responsible for keeping track of:

- the fees a user can claim
- the fees a trader can claim
- the fees an affiliate can claim
- the fees that should be deducted for a given pool
- the fee rebate for a user for a given pool

Only users that are registered in the Fees contract can hold LP tokens.

The Vault transfers the minted fees to this contract, which allocates it to the users who can claim their share.

Interactions with other contracts

Uses the `IPool` interface to access the creator (trader) of the pool, when a trader wants to claim their fee (or someone claims it for them).

- IPool::creator

Uses the `IERC20` interface to transfer the claimed fees and to query the balance of a pool.

- IERC20::balanceOf
- IERC20::safeTransfer

Uses the `IAffiliate` interface to get the commission levels, conversion ratio and claim limits for given ranks.

- IAffiliate::getCommissionLevelsForRanks
- IAffiliate::getLevelsAndConversionAndClaimLimitForRank

Uses the `IFeesClaimLimitAdapter` interface to cap the affiliate amounts.

- IFeesClaimLimitAdapter::onBeforeAffiliatePayout

Uses the `IUserProfile` interface to register users and get their parents.

- IUserProfile::getParentsAndParentRanks
- IUserProfile::getParentsAndBuyFeeDiscount
- IUserProfile::userExists
- IUserProfile::registerUserWoBooster
- IUserProfile::getParents

Issues

Medium

`_getCappedAffiliateAmount` depends on `IFeesClaimLimitAdapter` which is not yet implemented. The calls in lines 436, 333 and 278 potentially allow reentrency depending on the implementation of the interface. Function only makes sense, if `claimLimitAdapter` is not the null address. The function itself is not enforcing this, but all other callsite does.

Potential reentrancy in `_getParentsFromCache`. False positive.

```
Reentrancy in Fees._getParentsFromCache(address) (contracts/mosaic-alpha-contracts/Fees/Fees.sol#638-650):
    External calls:
    - IUserProfile(userProfile).registerUserWoBooster(_user) (contracts/mosaic-alpha-contracts/Fees/Fees.sol#642)
    State variables written after the call(s):
    - userFeeLevels[_user].parent = parent (contracts/mosaic-alpha-contracts/Fees/Fees.sol#644)
    - userFeeLevels[_user].parent2 = parent2 (contracts/mosaic-alpha-contracts/Fees/Fees.sol#645)
```

In:

- `Fees.onBeforeTransfer` ->

Out:

- ✓ `UserProfile.userExists`
- -> `UserProfile.registerUserWoBooster`
    - -> `AffiliateBooster.pickNextReferral`
        - ✓ `AffiliateBoosterController.epochClosed`
        - ✓ `AffiliateBoosterController.referralPicked`
    - ✓ `Affiliate.getLevelDetails`

- ✓ UserProfile.getParents

Potential reentrancy in onBeforeTransfer. False positive.

```
Reentrancy in Fees.onBeforeTransfer(address,address,address,uint256,uint256,uint256,uint256,address,IFees.OnBeforeTransferPayload)
(contracts/mosaic-alpha-contracts/Fees/Fees.sol#537-636):
    External calls:
    - (parentFrom,parentFrom2) = _getParentsFromCache(_from) (contracts/mosaic-alpha-contracts/Fees/Fees.sol#590)
        - IUserProfile(userProfile).registerUserWoBooster(_user) (contracts/mosaic-alpha-contracts/Fees/Fees.sol#642)
    - (parentTo,parentTo2) = _getParentsFromCache(_to) (contracts/mosaic-alpha-contracts/Fees/Fees.sol#617)
        - IUserProfile(userProfile).registerUserWoBooster(_user) (contracts/mosaic-alpha-contracts/Fees/Fees.sol#642)
    State variables written after the call(s):
    - (parentTo,parentTo2) = _getParentsFromCache(_to) (contracts/mosaic-alpha-contracts/Fees/Fees.sol#617)
        - userFeeLevels[_user].parentsCached = true (contracts/mosaic-alpha-contracts/Fees/Fees.sol#640)
        - userFeeLevels[_user].parent = parent (contracts/mosaic-alpha-contracts/Fees/Fees.sol#644)
        - userFeeLevels[_user].parent2 = parent2 (contracts/mosaic-alpha-contracts/Fees/Fees.sol#645)
    - _allocateUserL1ClaimableFees(_pool,parentTo) (contracts/mosaic-alpha-contracts/Fees/Fees.sol#621)
        - userPoolFeeStatus[_user][_pool].userClaimableL1Fee += uint128((l1Base * (claimablePerLp - claimablePerLpBefore)) >>
CLAIMABLE_PER_LP_MULTIPLIER_EXPONENT) (contracts/mosaic-alpha-contracts/Fees/Fees.sol#390)
        - userPoolFeeStatus[_user][_pool].lastClaimableAffiliateL1FeePerLp = claimablePerLp (contracts/mosaic-alpha-
contracts/Fees/Fees.sol#391)
    - _allocateUserL2ClaimableFees(_pool,parentTo2) (contracts/mosaic-alpha-contracts/Fees/Fees.sol#622)
        - userPoolFeeStatus[_user][_pool].userClaimableL2Fee += uint128((l2Base * (claimablePerLp - claimablePerLpBefore)) >>
CLAIMABLE_PER_LP_MULTIPLIER_EXPONENT) (contracts/mosaic-alpha-contracts/Fees/Fees.sol#403)
        - userPoolFeeStatus[_user][_pool].lastClaimableAffiliateL2FeePerLp = claimablePerLp (contracts/mosaic-alpha-
contracts/Fees/Fees.sol#404)
    - userPoolFeeStatus[parentFrom][_pool].l1Balance -= uint128(_amount) (contracts/mosaic-alpha-contracts/Fees/Fees.sol#628)
    - userPoolFeeStatus[parentTo][_pool].l1Balance += uint128(_amount) (contracts/mosaic-alpha-contracts/Fees/Fees.sol#629)
    - userPoolFeeStatus[parentFrom2][_pool].l2Balance -= uint128(_amount) (contracts/mosaic-alpha-contracts/Fees/Fees.sol#632)
    - userPoolFeeStatus[parentTo2][_pool].l2Balance += uint128(_amount) (contracts/mosaic-alpha-contracts/Fees/Fees.sol#633)
```

In:

- Fees.onBeforeTransfer ->

Out:

- ✓ UserProfile.userExists
- -> UserProfile.registerUserWoBooster
    - -> AffiliateBooster.pickNextReferral
        - ✓ AffiliateBoosterController.epochClosed
        - ✓ AffiliateBoosterController.referralPicked
    - ✓ Affiliate.getLevelDetails
- ✓ UserProfile.getParents

Potential reentrancy in selfManageMe. False positive.

```
Reentrancy in Fees.selfManageMe() (contracts/mosaic-alpha-contracts/Fees/Fees.sol#111-123):
    External calls:
    - _selfManageMeBefore() (contracts/mosaic-alpha-contracts/Fees/Fees.sol#115)
        - IUserProfile(userProfile).registerUser(slots[1]) (contracts/mosaic-alpha-contracts/Fees/Fees.sol#148)
    State variables written after the call(s):
    - governanceState.running = nextRunning (contracts/mosaic-alpha-contracts/Fees/Fees.sol#119)
    - governanceState.managers = IGovernance(governAddress).read_core_managers() (contracts/mosaic-alpha-
contracts/Fees/Fees.sol#120)
    - governanceState.governanceAddress = IGovernance(governAddress).read_core_govAddr() (contracts/mosaic-alpha-
contracts/Fees/Fees.sol#121)
```

In:

- Fees.selfManageMe ->

Out:

- -> UserProfile.registerUser
    - -> AffiliateBooster.pickNextReferral
        - ✓ AffiliateBoosterController.epochClosed
        - ✓ AffiliateBoosterController.referralPicked
    - ✓ Affiliate.getLevelDetails
- ✓ Governance.read_config_address_slot
- ✓ Governance.read_core_managers
- ✓ Governance.read_core_govAddr
- ✓ Governance.read_core_Running

FlashLoans.sol

Responsibilities

This contract is empty (only contains 2 empty methods).

Interactions with other contracts

Issues

Governance.sol

Responsibilities

Should be thought of as a key-value store for configs. The `Main` key contains the addresses of all contracts (this is used in the selfManageMe functions to set up communication between the contracts). There are a few further keys as well (e.g. `Fees`).

The Vault contract can't be replaced, but others can be for now if necessary. The addresses of the tokens can be changed here as well.

Each curator has a proportion with which they can vote for proposals. Proportions can be transferred among curators and the total sum of the proportions is constant.

See lib/test/MosaicDeployer.js for an example: we propose configs which are then voted on by curators. After this, the selfManageMe functions are called in batch which read the configs (for example the addresses of other contracts) and configure themselves accordingly.

Interactions with other contracts

Uses the `IGoverned` interface to call the `selfManageMe` function of the governed contracts.

- IGoverned::selfManageMe

Uses the `IVault` interface to call the `selfManageMe` function of the Vault when the vault config is updated.

- IVault::selfManageMe

Uses the `IAffiliate` interface to call the `selfManageMe` function of the Affiliate contract.

- IAffiliate::selfManageMe

Issues

Governed.sol

Responsibilities

Contains a default implementation of the `IGoverned` interface. Splits the `selfManageMe` function into 3 parts (`_selfManageMeBefore`, `_selfManageMeAfter`, `_onBeforeEmergencyChange`) to extract the common logic. Governed contracts inherit from this contract.

Interactions with other contracts

Uses the `IGovernance` interface to query information (is the core running, list of managers, governance address) from the Governance contract.

- IGovernance::read_core_Running
- IGovernance::read_core_managers
- IGovernance::read_core_govAddr

Issues

IERC20Burnable.sol

Responsibilities

Burnable IERC20 token interface from the OpenZeppelin library.

Interactions with other contracts

Issues

IFeesClaimLimitAdapter.sol

Responsibilities

Allows the affiliate system to define a limit for the amount of affiliate rewards that users of a given rank can claim in a month. Not implemented yet, but this interface leaves the option open.

Interactions with other contracts

Issues

IGoverned.sol

Responsibilities

Contracts that are handled by the governance system have to implement this interface.

The selfManageMe function is used for querying configs and self-configuration based on them.

Interactions with other contracts

Issues

Initializable.sol

Responsibilities

Helper contract to support initializer functions. To use it, replace the constructor with a function that has the `initializer` modifier.

Interactions with other contracts

Does not call other contracts.

Issues

IQueryApi.sol

Responsibilities

Groups read-only queries into a single transaction for the backend.

Interactions with other contracts

Issues

IRegister.sol

Responsibilities

Interface for the trader whitelist. See `Register.sol`

Interactions with other contracts

Issues

Math.sol

Responsibilities

A helper contract for fixed point calculations.

Interactions with other contracts

This contract doesn't call other contracts.

Issues

There are no tests that target this library. It is only tested indirectly.

Ownable.sol

Responsibilities

Contract module which provides a basic access control mechanism, where there is an account (an owner) that can be granted exclusive access to specific functions.

By default, the owner account will be the one that deploys the contract. This can later be changed with `transferOwnership`.

This module is used through inheritance. It will make available the modifier `onlyOwner`, which can be applied to your functions to restrict their use to the owner.

This contract is from the OpenZeppelin library.

Interactions with other contracts

Does not call other contracts.

Issues

Pool.sol

Responsibilities

This contract represents the baskets that can be traded on the mosaic alpha platform. Every time a trader creates a new basket, this contract is cloned by the PoolFactory.

There are 3 types of pools defined:

- REBALANCING: (eg. 30% ETH, 30% BTC, 40% MKR). Weight changes gradually with time.
- NON_REBALANCING: (eg. 100 ETH, 5 BTC, 200 MKR). Weight changes gradually with time.
- DAYTRADE: Non rebalancing pool. Weight changes immediately.

Interactions with other contracts

Uses the `IFees` interface to check if fees are valid and receives an IFees.MosaicPoolFees instance to access its buyFee member

- IFees::MosaicPoolFees
- IFees::isValidPoolFees
- IFees::isValidBuyFee

Uses the `IVault` interface to initialize the pool, disable the pool, to implement the IERC20 interface methods. (Too many method calls to list everything here but the list below is complete.)

- IVault::disableFees
- IVault::poolAddressToId
- IVault::registerTokens
- IVault::getVaultState
- IVault::setPoolEmergencyMode
- IVault::getInternalBalance
- IVault::transferFromAsTokenContract
- IVault::getPoolTotalSupplyBaseByAddr
- IVault::TotalSupplyBase
- IVault::tokenInPool
- IVault::deregisterToken
- IVault::isRunning
- IVault::getAuctionInfo
- IVault::isTokenWhitelisted
- IVault::startAuction
- IVault::stopAuction

Issues

Medium

Potential reentrancy in `dutchAuction`. False positive.

```
Reentrancy in Pool.dutchAuction(address,uint256,address,uint32,uint32,uint256) (contracts/mosaic-alpha-
contracts/Pool/Pool.sol#627-655):
    External calls:
    - _auctionId = IVault(vaultAddress).startAuction(tokenToSell,amountToSell,tokenToBuy,duration,expiration,endingPrice)
(contracts/mosaic-alpha-contracts/Pool/Pool.sol#648)
    State variables written after the call(s):
    - runningDutchAuctionIds.push(_auctionId) (contracts/mosaic-alpha-contracts/Pool/Pool.sol#651)
```

In:

- `Pool.dutchAuction` -> `onlyPoolOwnerOrManager`

Out:

- ✓ `Vault.isRunning`
- ✓ `Vault.getAuctionInfo`
- ✓ `Vault.tokenInPool`
- -> `Vault.startAuction`
    - -> `SimpleMosaicOracle.getBidPrice`
    - ✓ `PancakePair.token0`
    - ✓ `PancakePair.getReserves`
    - ✓ `PancakeFactory.getPair`
    - ✓ `PancakeRouter02.factory`
- ✓ `Vault.isTokenWhitelisted`

Potential reentrancy in `removeToken`. False positive.

```
Reentrancy in Pool.removeToken(uint256) (contracts/mosaic-alpha-contracts/Pool/Pool.sol#569-590):
    External calls:
    - IVault(vaultAddress).deregisterToken(tokens[_index],_remainingBalance) (contracts/mosaic-alpha-contracts/Pool/Pool.sol#581)
    State variables written after the call(s):
    - tokens[_index] = tokens[tokens.length - 1] (contracts/mosaic-alpha-contracts/Pool/Pool.sol#583)
    - tokens.pop() (contracts/mosaic-alpha-contracts/Pool/Pool.sol#584)
```

In:

- `Pool.removeToken` -> `onlyPoolOwnerOrManager`

Out:

- ✓ `Vault.isRunning`
- -> `Vault.deregisterToken`
    - -> `Fees.onBeforeTransfer`
    - ✓ `UserProfile.userExists`
    - -> `UserProfile.registerUserWoBooster`
        - -> `AffiliateBooster.pickNextReferral`
            - ✓ `AffiliateBoosterController.epochClosed`
            - ✓ `AffiliateBoosterController.referralPicked`
        - ✓ `Affiliate.getLevelDetails`
    - ✓ `UserProfile.getParents`
    - -> `SimpleMosaicOracle.consultUsdPoolPricePerLp`
    - -> `Vault.getPoolTokensByAddr`
        - ✓ `Pool.getTokens`
    - ✓ `Pool.getTrailingFeeAmount`
    - ✓ `PancakePair.token0`
    - ✓ `PancakePair.getReserves`
    - ✓ `PancakeFactory.getPair`
    - ✓ `PancakeRouter02.factory`
    - ✓ `Pool.updatePerfFeePricePerLp`
    - ✓ `Pool.getTrailingFeeAmount`
    - ✓ `Pool.creator`
    - ✓ `Pool.emitTransfer`
- ✓ `Vault.getInternalBalance`

PoolFactory.sol

Responsibilities

Can create new pools initiated by traders. Clones the Pool contract and sets the new pool's initialy liqudity in the Vault.

Interactions with other contracts

Passes an `IFees.MosaicPoolFees`instance to Pool::initialize.

Clones the `Pool` contract and calls its initalize function

- Pool::initialize

Uses the `IRegister` interface to query if the user initiating the pool creation is an approved trader.

- IRegister::isApprovedTrader

Uses the `IVault` interface to register the new pool and to transfer tokens to it.

- IVault::registerPool
- IVault::depositToInternalBalance
- IVault::transferInternalBalance
- IVault::addInitialLiquidity

Uses the `IFeeBurner` interface to deduct pool creation fee and burn it.

- IFeeBurner::feeToken
- IFeeBurner::burn

Uses the `IERC20` interface to approve transfers and transfer tokens.

- IERC20::transferFrom
- IERC20::approve

Issues

The `create` function depends on the tokens' `transferFrom` (line 196) and `approve` (line 203) functions always returning `true`. Should any of them return with `false`, the tokens might remain with the user or one of the contracts.

PoolLibrary.sol

Responsibilities

Isn't used, should be ignored.

Interactions with other contracts

Does not call other contracts.

Issues

QueryApi.sol

Responsibilities

A helper contract that allows easy querying of pool data in batch. (Used by the backend to avoid having to make multiple transactions).

Interactions with other contracts

Uses the `IVault` interface to query the number of pools in the vault and to get the addresses of the pools and their swap fees.

- IVault::poolCount
- IVault::poolIdToAddress
- IVault::getVaultState

Uses the `IPool` interface to get a pool's attributes

- IPool::name
- IPool::owner
- IPool::poolType
- IPool::isUnlocked
- IPool::getTokens
- IPool::getWeights
- IPool::getReserves
- IPool::getPoolFees

Issues

Register.sol

Responsibilities

Responsible for keeping track of approved pools and traders, and of blacklisted and whitelisted addresses.

Interactions with other contracts

Uses the `IVault` interface to convert between pool IDs and addresses.

- IVault::poolIdToAddress
- IVault::poolAddressToId

Issues

SimpleAffiliateBoosterController.sol

Responsibilities

Organizes the pricing logic of the AffiliateBooster into a separate contract to make it replaceable. Allows the owner to set the ticket price and fee token.

Interactions with other contracts

Uses the `IERC20Burnable` interface to burn (or redirect to a given address if not burnable) the price of purchased tickets (in the fee token).

- IERC20Burnable::transferFrom
- IERC20Burnable::burn
- IERC20Burnable::transfer

Uses the `IAffiliateBooster` interface to allocate a ticket when a user puchases one.

- IAffiliateBooster::allocateTicket

Issues

SimpleDeposit.sol

Responsibilities

This is the contract responsible for staking. Users can deposit and withdraw KDX and lock (stake) and unlock (unstake) their deposited KDX.

Interactions with other contracts

Uses the `IFees` interface to notify the `Fees` contract about a changed staking.

- IFees::userStakeChanged

Uses the `IAffiliate` interface to notify the `Affiliate` contract about a changed staking.

- IAffiliate::userStakeChanged

Uses the `IRegister` interface to query if a user is whitelisted.

- IRegister::isWhitelisted

Uses the `IERC20` interface to transfer tokens to itself when a deposit is made and from itself when staking is withdrawn.

- IERC20::transferFrom
- IERC20::transfer

Issues

SimpleEntropy.sol

Responsibilities

This contract is used for random number generation.

Interactions with other contracts

Does not call other contracts.

Issues

SimpleMosaicOracle.sol

Responsibilities

This contract is an oracle for getting prices.

Interactions with other contracts

Uses the `IVault` interface to get the tokens in a pool to calculate the USDT price of an LP token and receives an IVault.TotalSupplyBase instance as a function parameter

- IVault::getPoolTokensByAddr

Uses the `IPool` interface to get the trailing fee of a pool

- IPool::getTrailingFeeAmount

Issues

Swaps.sol

Responsibilities

Offers LP tokens in exchange for USDT: the user transfers their USDT to the Swaps contract, which buys tokens from the Pancake swap (after querying the pool for its token ratio), then trades the tokens for LP tokens using the Vault contract and returns them to the user.

Interactions with other contracts

Uses the `IVault` interface to mint and burn LP tokens and to deposit tokens to a user's internal balance.

- IVault::Mint
- IVault::Burn
- IVault::depositToInternalBalance
- IVault::getInternalBalance
- IVault::getVaultState

Uses the `IERC20` interface to transfer USDT to itself and to transfer the remaining USDT back.

- IERC20::approve
- IERC20::transferFrom
- IERC20::balanceOf
- IERC20::safeTransfer
- IERC20::transfer

Uses the `IPool` interface to get the amounts of tokens needed to mint an LP token and to get the amounts of tokens received upon burning an LP token, as well as to get the tokens of a pool.

- IPool::calcMintAmounts
- IPool::getTokens
- IPool::calcBurnAmounts
- IPool::poolId
- IPool::queryExactTokensForTokens
- IPool::queryTokensForExactTokens

Uses the `IPancakeRouter02` interface to communicate with the Pancake swap.

- IPancakeRouter02::getAmountsOut
- IPancakeRouter02::getAmountsIn
- IPancakeRouter02::swapTokensForExactTokens

Issues

Functions `addTokens` and `removeTokens` are subject to reentrancy via `_approveTokens` and `IERC20`. Not sure if it can be used maliciously or not.

SystemRole.sol

Responsibilities

This contract is used for role control of other contracts There are 4 possible roles:

- ADMIN
- SYSTEM_ADMIN
- TRADER
- INVESTOR

ADMIN role can add (setup) or remove (revoke) roles to addresses. For every role there is modifier defined to restrict function access.

Interactions with other contracts

Does not call other contracts.

Issues

SystemWhitelist.sol

Responsibilities

This contract is used for whitelisting and blacklisting addresses.

Interactions with other contracts

Does not call other contracts.

Issues

UserProfile.sol

Responsibilities

Keeps track of referral relationships and user data (for example the permanent buy fee discount a user is entitled to based on their referrer) as well as the user ranks.

Only users that are registered in the UserProfile contract can hold LP tokens.

Interactions with other contracts

Uses the `IAffiliateBooster` interface to get the referrer of a user.

- IAffiliateBooster::pickNextReferral

Uses the `IAffiliate` interface to get the buy fee discount for a given discount level.

- IAffiliate::getLevelDetails

Uses the `IFees` interface to notify the Fees contract of a user rank change.

- IFees::userRankChanged

Uses the `Address` contract to decide if an address is a contract.

- Address::isContract

Issues

Medium

Potential reentrancy in `_registerUser`. This issue is also reported at `AffiliateBooster`. False positive.

```
Reentrancy in UserProfile._registerUser(address,address) (contracts/mosaic-alpha-contracts/User/UserProfile.sol#226-247):
    External calls:
    - _referredBy = IAffiliateBooster(affiliateBoosterContract).pickNextReferral() (contracts/mosaic-alpha-
contracts/User/UserProfile.sol#230)
    - _registerUser(_referredBy,address(0)) (contracts/mosaic-alpha-contracts/User/UserProfile.sol#236)
        - _referredBy = IAffiliateBooster(affiliateBoosterContract).pickNextReferral() (contracts/mosaic-alpha-
contracts/User/UserProfile.sol#230)
    State variables written after the call(s):
    - userCount ++ (contracts/mosaic-alpha-contracts/User/UserProfile.sol#237)
    - userReferrals[_referredBy].push(_user) (contracts/mosaic-alpha-contracts/User/UserProfile.sol#244)
    - _registerUser(_referredBy,address(0)) (contracts/mosaic-alpha-contracts/User/UserProfile.sol#236)
        - users[_user].exists = true (contracts/mosaic-alpha-contracts/User/UserProfile.sol#238)
        - users[_user].referredBy = _referredBy (contracts/mosaic-alpha-contracts/User/UserProfile.sol#239)
        - users[_user].referredByBefore = users[_referredBy].referredBy (contracts/mosaic-alpha-
contracts/User/UserProfile.sol#240)
        - users[_user].referredByRank = users[_referredBy].rank (contracts/mosaic-alpha-contracts/User/UserProfile.sol#241)
        - users[_user].buyFeeDiscount =
IAffiliate(affiliateContract).getLevelDetails(users[_referredBy].rank).referralBuyFeeDiscount (contracts/mosaic-alpha-
contracts/User/UserProfile.sol#242)
        - users[_referredBy].referralCount ++ (contracts/mosaic-alpha-contracts/User/UserProfile.sol#243)
    - users[_user].exists = true (contracts/mosaic-alpha-contracts/User/UserProfile.sol#238)
    - users[_user].referredBy = _referredBy (contracts/mosaic-alpha-contracts/User/UserProfile.sol#239)
    - users[_user].referredByBefore = users[_referredBy].referredBy (contracts/mosaic-alpha-contracts/User/UserProfile.sol#240)
    - users[_user].referredByRank = users[_referredBy].rank (contracts/mosaic-alpha-contracts/User/UserProfile.sol#241)
    - users[_user].buyFeeDiscount = IAffiliate(affiliateContract).getLevelDetails(users[_referredBy].rank).referralBuyFeeDiscount
(contracts/mosaic-alpha-contracts/User/UserProfile.sol#242)
    - users[_referredBy].referralCount ++ (contracts/mosaic-alpha-contracts/User/UserProfile.sol#243)
```

In:

- `UserProfile.setUserRank` ->
- `UserProfile.registerUser` ->
- `UserProfile.registerUserWoBooster` -> `onlyController`

Out:

- -> `AffiliateBooster.pickNextReferral`
    - ✓ `AffiliateBoosterController.epochClosed`
    - ✓ `AffiliateBoosterController.referralPicked`
- ✓ `Affiliate.getLevelDetails`

Vault.sol

Responsibilities

Responsible for storing the internal balance. The Vault is the only contract that holds the underlying tokens.

The internal balance is an address -> address -> uint mapping (this user own this much from this token). Contains the balances of the pools as well (trader creates new pool -> PoolFactory clones the Pool contract, then sets its initial liquidity in the Vault).

The Vault is responsible for issuing LP tokens in exchange for the underlying tokens (bitcoin, eth, etc.) and also in the other direction (the former is called minting and the latter is called burning).

Has a function that can be called by anyone to trigger the performance and trailing fee calculation and deduction.

The Vault is also responsible for minting the LP tokens for the fees (which are "deducted" through inflation and redistribution). The minted fee LPs get transferred to the Fees contract, which is resposnsible for its redistribution.

The Vault is also responsible for keeping the token whitelist.

Interactions with other contracts

Uses the `IPool` interface to get metadata about pools, emit transfer events, get the token list and mint and burn fee tokens.

- IPool::emitTransfer
- IPool::getTrailingFeeAmount
- IPool::updatePerfFeePricePerLp
- IPool::creator
- IPool::getTokens
- IPool::owner
- IPool::initialLiquidityProvided
- IPool::_mint
- IPool::_burn
- IPool::queryExactTokensForTokens
- IPool::queryTokensForExactTokens

Uses the `IUserProfile` interface to register users.

- IUserProfile::registerUser

Uses the `IAffiliate` interface to register user purchases.

- IAffiliate::registerUserPurchase
- IAffiliate::registerUserPurchaseAsTokens

Uses the `IFees` interface to allocate fees, and passes an IFees.OnBeforeTransferPayload to IFees::onBeforeTransfer.

- IFees::onBeforeTransfer
- IFees::allocateTrailingFee
- IFees::allocatePerformanceFee
- IFees::allocateBuyFee

Uses the `IRegister` interface to query whether a user is whitelisted.

- IRegister::isWhitelisted

Uses the `IMosaicOracle` interface to query the USDT prices of tokens and to get the starting price for biddings.

- IMosaicOracle::consultUsdPoolPricePerLp
- IMosaicOracle::getUsdPoolPrice
- IMosaicOracle::getBidPrice

Uses the `IERC20` interface to transfer to and from the internal balance, to transfer tokens in case of an emergency withdrawal and to provide flash loans.

- IERC20::safeTransferFrom
- IERC20::safeTransfer
- IERC20::balanceOf
- IERC20::transfer
- IERC20::transferFrom

Issues

High

Unchecked transfer in line 1367. `transfer` might return `false`. Probably non-issue, as it can be executed again.

Suspicious code in lines 608-610. Unnecessary `if`, TODO comment.

Medium

Potential reentrancy in `_burnFromInternalBalance`. False positive.

```
Reentrancy in Vault._burnFromInternalBalance(address,address,uint256,bool,bool) (contracts/mosaic-alpha-
contracts/Vault/Vault.sol#526-537):
    External calls:
    - _onBeforeBalanceTransfer(token,from,address(0),amount,feesAlreadyAllocated,true,false) (contracts/mosaic-alpha-
contracts/Vault/Vault.sol#529)
        - IMosaicOracle(mosaicOracle).consultUsdPoolPricePerLp(poolAddr,ts) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#703)
        - IFees(fees).onBeforeTransfer(_pool,_from,_to,_internalBalance[_from][_pool],_internalBalance[_to]
[_pool],amount,originalTotalsupplyBase,trader,payload) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#598-601)
        - performanceFeesToMint = (IPool(_pool).updatePerfFeePricePerLp(tsTmp,valuePerLp)) (contracts/mosaic-alpha-
contracts/Vault/Vault.sol#568)
        - IPool(_pool).emitTransfer(address(0),fees,amount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#610)
        - IFees(fees).onBeforeTransfer(_pool,_from,_to,_internalBalance[_from][_pool],_internalBalance[_to]
[_pool],amount,0,address(0),payload) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#614)
    State variables written after the call(s):
    - _internalBalance[from][token] -= amount (contracts/mosaic-alpha-contracts/Vault/Vault.sol#530)
    - _handlerForPoolOfUser(from,token,false) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#534)
        - poolOfUser[user].push(poolId) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1226)
        - poolOfUser[user][index_scope_0] = replacementPoolId (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1240)
        - poolOfUser[user].pop() (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1241)
    - _handlerForPoolOfUser(from,token,false) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#534)
        - poolOfUserIndexes[user][poolId] = (poolOfUser[user].length).toUint32() (contracts/mosaic-alpha-
contracts/Vault/Vault.sol#1228)
        - poolOfUserIndexes[user][replacementPoolId] = uint32(index_scope_0) (contracts/mosaic-alpha-
contracts/Vault/Vault.sol#1239)
        - poolOfUserIndexes[user][poolId] = 0 (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1242)
    - poolStates[token].totalSupplyBase.amount -= (amount).toUint224() (contracts/mosaic-alpha-contracts/Vault/Vault.sol#531)
    - poolStates[token].totalSupplyBase.timestamp = uint32(block.timestamp) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#532)
```

In:

- `Vault.burnAsTokenContract` ->
- `Vault.Burn` ->

Out:

- -> `Fees.onBeforeTransfer`
- ✓ `UserProfile.userExists`
- -> `UserProfile.registerUserWoBooster`
    - -> `AffiliateBooster.pickNextReferral`
        - ✓ `AffiliateBoosterController.epochClosed`
        - ✓ `AffiliateBoosterController.referralPicked`
    - ✓ `Affiliate.getLevelDetails`
- ✓ `UserProfile.getParents`
- -> `SimpleMosaicOracle.consultUsdPoolPricePerLp`
- -> `Vault.getPoolTokensByAddr`
    - ✓ `Pool.getTokens`
- ✓ `Pool.getTrailingFeeAmount`
- ✓ `PancakePair.token0`
- ✓ `PancakePair.getReserves`
- ✓ `PancakeFactory.getPair`
- ✓ `PancakeRouter02.factory`
- ✓ `Pool.updatePerfFeePricePerLp`
- ✓ `Pool.getTrailingFeeAmount`
- ✓ `Pool.creator`
- ✓ `Pool.emitTransfer`

Potential reentrancy in `_mintToInternalBalance`. False positive.

```
Reentrancy in Vault._mintToInternalBalance(address,address,uint256,bool,bool) (contracts/mosaic-alpha-
contracts/Vault/Vault.sol#510-522):
    External calls:
    - _onBeforeBalanceTransfer(token,address(0),to,amount,feesAlreadyAllocated,true,false) (contracts/mosaic-alpha-
contracts/Vault/Vault.sol#513)
        - IMosaicOracle(mosaicOracle).consultUsdPoolPricePerLp(poolAddr,ts) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#703)
        - IFees(fees).onBeforeTransfer(_pool,_from,_to,_internalBalance[_from][_pool],_internalBalance[_to]
[_pool],amount,originalTotalsupplyBase,trader,payload) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#598-601)
        - performanceFeesToMint = (IPool(_pool).updatePerfFeePricePerLp(tsTmp,valuePerLp)) (contracts/mosaic-alpha-
contracts/Vault/Vault.sol#568)
        - IPool(_pool).emitTransfer(address(0),fees,amount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#610)
        - IFees(fees).onBeforeTransfer(_pool,_from,_to,_internalBalance[_from][_pool],_internalBalance[_to]
[_pool],amount,0,address(0),payload) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#614)
    State variables written after the call(s):
    - _internalBalance[to][token] += amount (contracts/mosaic-alpha-contracts/Vault/Vault.sol#517)
    - _handlerForPoolOfUser(to,token,true) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#518)
        - poolOfUser[user].push(poolId) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1226)
        - poolOfUser[user][index_scope_0] = replacementPoolId (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1240)
        - poolOfUser[user].pop() (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1241)
    - _handlerForPoolOfUser(to,token,true) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#518)
        - poolOfUserIndexes[user][poolId] = (poolOfUser[user].length).toUint32() (contracts/mosaic-alpha-
contracts/Vault/Vault.sol#1228)
        - poolOfUserIndexes[user][replacementPoolId] = uint32(index_scope_0) (contracts/mosaic-alpha-
```

```
contracts/Vault/Vault.sol#1239)
        - poolOfUserIndexes[user][poolId] = 0 (contracts/mosaic-alpha-contracts/Vault/Vault.sol#1242)
    - poolStates[token].totalSupplyBase.amount += (amount).toUint224() (contracts/mosaic-alpha-contracts/Vault/Vault.sol#514)
    - poolStates[token].totalSupplyBase.timestamp = uint32(block.timestamp) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#515)
```

In

- `Vault.mintAsTokenContract` ->
- `Vault.addInitialLiquidity` ->
- `Vault.Mint` ->
- `Vault.Burn` ->

Out

- -> `Fees.onBeforeTransfer`
- ✓ `UserProfile.userExists`
- -> `UserProfile.registerUserWoBooster`
    - -> `AffiliateBooster.pickNextReferral`
        - ✓ `AffiliateBoosterController.epochClosed`
        - ✓ `AffiliateBoosterController.referralPicked`
    - ✓ `Affiliate.getLevelDetails`
- ✓ `UserProfile.getParents`
- -> `SimpleMosaicOracle.consultUsdPoolPricePerLp`
- -> `Vault.getPoolTokensByAddr`
    - ✓ `Pool.getTokens`
- ✓ `Pool.getTrailingFeeAmount`
- ✓ `PancakePair.token0`
- ✓ `PancakePair.getReserves`
- ✓ `PancakeFactory.getPair`
- ✓ `PancakeRouter02.factory`
- ✓ `Pool.updatePerfFeePricePerLp`
- ✓ `Pool.getTrailingFeeAmount`
- ✓ `Pool.creator`
- ✓ `Pool.emitTransfer`

Potential reentrancy in `_calculateTrailingAndPerformanceFee`. False positive.

```
Reentrancy in Vault._calculateTrailingAndPerformanceFee(address,bool,bool) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#544-
572):
    External calls:
    - valuePerLp = _consultPoolPricePerLpByAddr(_pool,tsTmp) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#566)
        - IMosaicOracle(mosaicOracle).consultUsdPoolPricePerLp(poolAddr,ts) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#703)
    - performanceFeesToMint = (IPool(_pool).updatePerfFeePricePerLp(tsTmp,valuePerLp)) (contracts/mosaic-alpha-
contracts/Vault/Vault.sol#568)
    State variables written after the call(s):
    - poolStates[_pool].lastPerformanceTimestamp = uint48(block.timestamp) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#570)
```

In:

- `Vault.emergenyWithdrawPool` -> `onlyAdmin`
- `Vault.multiSwap` ->
- `Vault.transferFromAsTokenContract` ->
- `Vault.mintAsTokenContract` ->
- `Vault.burnAsTokenContract` ->
- `Vault.deregisterToken` ->
- `Vault.addInitialLiquidity` ->
- `Vault.transferInternalBalance` ->
- `Vault.bid` ->
- `Vault.Mint` ->
- `Vault.emergenyWithdraw` ->
- `Vault.Burn` ->

Out:

- -> `SimpleMosaicOracle.consultUsdPoolPricePerLp`
- -> `Vault.getPoolTokensByAddr`
    - ✓ `Pool.getTokens`
- ✓ `Pool.getTrailingFeeAmount`
- ✓ `PancakePair.token0`
- ✓ `PancakePair.getReserves`
- ✓ `PancakeFactory.getPair`
- ✓ `PancakeRouter02.factory`
- ✓ `Pool.updatePerfFeePricePerLp`
- ✓ `Pool.getTrailingFeeAmount`

Potential reentrancy in `_onBeforeBalanceTransfer`. Callgraph needs to be explored. Seems to be a common point between the potential issues in `_burnFromInternalBalance` and `_mintToInternalBalance`. False positive.

```
Reentrancy in Vault._onBeforeBalanceTransfer(address,address,address,uint256,bool,bool,bool) (contracts/mosaic-alpha-
contracts/Vault/Vault.sol#583-615):
    External calls:
    - (payload.trailingLpToMint,payload.performanceLpToMint) =
```

```
_calculateTrailingAndPerformanceFee(_pool,forceTrailingFeeAllocation,forcePerformanceFeeAllocation) (contracts/mosaic-alpha-
contracts/Vault/Vault.sol#591)
        - IMosaicOracle(mosaicOracle).consultUsdPoolPricePerLp(poolAddr,ts) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#703)
        - performanceFeesToMint = (IPool(_pool).updatePerfFeePricePerLp(tsTmp,valuePerLp)) (contracts/mosaic-alpha-
contracts/Vault/Vault.sol#568)
    - IFees(fees).onBeforeTransfer(_pool,_from,_to,_internalBalance[_from][_pool],_internalBalance[_to]
[_pool],amount,originalTotalsupplyBase,trader,payload) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#598-601)
    State variables written after the call(s):
    - _internalBalance[fees][_pool] += expAmount (contracts/mosaic-alpha-contracts/Vault/Vault.sol#606)
    - poolStates[_pool].totalSupplyBase.amount += uint224(expAmount) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#603)
    - poolStates[_pool].totalSupplyBase.timestamp = (block.timestamp).toUint32() (contracts/mosaic-alpha-
contracts/Vault/Vault.sol#604)
```

In:

- `Vault.emergenyWithdrawPool` -> `onlyAdmin`
- `Vault.multiSwap` ->
- `Vault.transferFromAsTokenContract` ->
- `Vault.mintAsTokenContract` ->
- `Vault.burnAsTokenContract` ->
- `Vault.deregisterToken` ->
- `Vault.addInitialLiquidity` ->
- `Vault.transferInternalBalance` ->
- `Vault.bid` ->
- `Vault.Mint` ->
- `Vault.emergenyWithdraw` ->
- `Vault.Burn` ->

Out:

- -> `Fees.onBeforeTransfer`
- ✓ `UserProfile.userExists`
- -> `UserProfile.registerUserWoBooster`
    - -> `AffiliateBooster.pickNextReferral`
        - ✓ `AffiliateBoosterController.epochClosed`
        - ✓ `AffiliateBoosterController.referralPicked`
    - ✓ `Affiliate.getLevelDetails`
- ✓ `UserProfile.getParents`
- -> `SimpleMosaicOracle.consultUsdPoolPricePerLp`
- -> `Vault.getPoolTokensByAddr`
    - ✓ `Pool.getTokens`
- ✓ `Pool.getTrailingFeeAmount`
- ✓ `PancakePair.token0`
- ✓ `PancakePair.getReserves`
- ✓ `PancakeFactory.getPair`
- ✓ `PancakeRouter02.factory`
- ✓ `Pool.updatePerfFeePricePerLp`
- ✓ `Pool.getTrailingFeeAmount`
- ✓ `Pool.creator`
- ✓ `Pool.emitTransfer`

Potential reentrancy in `registerPool`. False positive.

```
Reentrancy in Vault.registerPool(address,address,address) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#721-739):
    External calls:
    - _registerUser(_user,_referredBy) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#727)
        - IUserProfile(userProfile).registerUser(to,referredBy) (contracts/mosaic-alpha-contracts/Vault/Vault.sol#202)
    State variables written after the call(s):
    - _poolAddressToId[_pool] = _poolId (contracts/mosaic-alpha-contracts/Vault/Vault.sol#732)
```

In:

- `Vault.registerPool` -> `onlyFactory`

Out:

- -> `UserProfile.registerUser`
    - -> `AffiliateBooster.pickNextReferral`
        - ✓ `AffiliateBoosterController.epochClosed`
        - ✓ `AffiliateBoosterController.referralPicked`
    - ✓ `Affiliate.getLevelDetails`