// HALBORN

0x_nodes

Smart Contract Security Audit

Prepared by: **Halborn**

Date of Engagement: **September 25th, 2021 - October 12th, 2021**

Visit: **Halborn.com**

DOCUMENT REVISION HISTORY

VERSION	MODIFICATION	DATE	AUTHOR
0.1	Document Creation	09/25/2021	Timur Guvenkaya
0.2	Document Edit	09/27/2021	Timur Guvenkaya
0.3	Document Edit	10/03/2021	Timur Guvenkaya
0.4	Document Edit	10/07/2021	Timur Guvenkaya
0.5	Final Draft	10/12/2021	Timur Guvenkaya
0.6	Final Draft Review	10/13/2021	Gabi Urrutia
1.0	Remediation Plan Edit	11/07/2021	Timur Guvenkaya
1.1	Remediation Plan Edit	11/12/2021	Timur Guvenkaya
1.1	Remediation Plan Review	11/17/2021	Gabi Urrutia

CONTACTS

CONTACT	COMPANY	EMAIL
Rob Behnke	Halborn	Rob.Behnke@halborn.com
Steven Walbroehl	Halborn	Steven.Walbroehl@halborn.com
Gabi Urrutia	Halborn	Gabi.Urrutia@halborn.com
Timur Guvenkaya	Halborn	Timur.Guvenkaya@halborn.com

EXECUTIVE OVERVIEW

1.1 INTRODUCTION

0x_nodes engaged Halborn to conduct a security assessment on their smart contracts beginning on September 25th, 2021 and ending October 12th, 2021. 0x_nodes is a system that allows users to simplify access to defi yields. Users deposit assets to the system, the assets are put into various yield-generating strategies, and the users collect rewards.

Though this security audit's outcome is satisfactory, only the most essential aspects were tested and verified to achieve objectives and deliverables set in the scope due to time and resource constraints. It is essential to note the use of the best practices for secure development.

1.2 AUDIT SUMMARY

The team at Halborn was provided four weeks for the engagement and assigned one full time security engineer to audit the security of the assets in scope. The engineer is a blockchain and smart contract security expert with advanced penetration testing, smart-contract hacking, and deep knowledge of multiple blockchain protocols.

The purpose of this audit is to achieve the following:

- Identify potential security issues with the smart contracts.

In summary, Halborn identified few security risks that should be addressed by 0x_nodes team.

1.3 TEST APPROACH & METHODOLOGY

Halborn performed a combination of manual and automated security testing to balance efficiency, timeliness, practicality, and accuracy in regard to the scope of the smart contract audit. While manual testing is recommended

to uncover flaws in logic, process,and implementation; automated testing techniques help enhance coverage of the smart contract code and can quickly identify items that do not follow security best practices. The following phases and associated tools were used throughout the term of the audit:

- Research into architecture and purpose.
- Smart contract manual code review and walkthrough
- Graphing out functionality and contract logic/connectivity/functions(solgraph)
- Manual testing of core functions through Hardhat and Ganache
- Manual testing with custom scripts.
- Static Analysis of security for scoped contract, and imported functions.(Slither)
- Scanning of solidity files for vulnerabilities, security hotspots or bugs. (MythX)
- Testnet deployment (Remix IDE)

RISK METHODOLOGY:

Vulnerabilities or issues observed by Halborn are ranked based on the risk assessment methodology by measuring the **LIKELIHOOD** of a security incident, and the **IMPACT** should an incident occur. This framework works for communicating the characteristics and impacts of technology vulnerabilities. It's quantitative model ensures repeatable and accurate measurement while enabling users to see the underlying vulnerability characteristics that was used to generate the Risk scores. For every vulnerability, a risk level will be calculated on a scale of 5 to 1 with 5 being the highest likelihood or impact.

RISK SCALE - LIKELIHOOD

5 - Almost certain an incident will occur.
4 - High probability of an incident occurring.
3 - Potential of a security incident in the long term.
2 - Low probability of an incident occurring.
1 - Very unlikely issue will cause an incident.

RISK SCALE - IMPACT

5 - May cause devastating and unrecoverable impact or loss.
4 - May cause a significant level of impact or loss.
3 - May cause a partial impact or loss to many.
2 - May cause temporary impact or loss.
1 - May cause minimal or un-noticeable impact.

The risk level is then calculated using a sum of these two values, creating a value of 10 to 1 with 10 being the highest level of security risk.

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL

10 - CRITICAL
9 - 8 - HIGH
7 - 6 - MEDIUM
5 - 4 - LOW
3 - 1 - VERY LOW AND INFORMATIONAL

1.4 SCOPE

The review was scoped to contracts and deployment scripts in the audit branch in platform repository.

- Smart Contracts
  - BiosRewards.sol
  - Controlled.sol
  - EtherRewards.sol
  - IntegrationMap.sol
  - Kernel.sol
  - ModuleMap.sol
  - ModuleMapConsumer.sol
  - StrategyManager.sol
  - StrategyMap.sol
  - SushiSwapTrader.sol
  - UniswapTrader.sol
  - UserPositions.sol
  - YieldManager.sol
  - AaveIntegration.sol
  - DynamicRangeOrdersIntegration.sol
  - DynamicRangeOrdersIntegrationDeployer.sol
  - SushiSwapIntegration.sol
  - UniswapIntegration.sol
  - UniswapIntegrationDeployer.sol
  - YearnIntegration.sol

OUT-OF-SCOPE:
Other smart contracts in the repository, external libraries and economical attacks.

2. ASSESSMENT SUMMARY & FINDINGS OVERVIEW

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL
0	0	2	13	6

LIKELIHOOD

IMPACT

(HAL-03)	(HAL-01)			
	(HAL-04) (HAL-05) (HAL-06)	(HAL-02)		
	(HAL-10) (HAL-11) (HAL-12) (HAL-13)	(HAL-07) (HAL-08) (HAL-09)		
(HAL-18) (HAL-19) (HAL-20) (HAL-21)	(HAL-16) (HAL-17)	(HAL-14) (HAL-15)		

SECURITY ANALYSIS	RISK LEVEL	REMEDIATION DATE
IMPROPER APPLICATION OF PRINCIPLE OF LEAST PRIVILEGE	Medium	RISK ACCEPTED
MISSING RE-ENTRANCY PROTECTION	Medium	SOLVED - 10/26/2021
LACK OF PAUSE FUNCTION	Low	ACKNOWLEDGED
DIVIDE BEFORE MULTIPLY	Low	SOLVED - 10/26/2021
USE OF BLOCK.TIMESTAMP	Low	NOT APPLICABLE
EXTERNAL FUNCTION CALLS WITHIN LOOP	Low	ACKNOWLEDGED
IGNORED RETURN VALUES	Low	ACKNOWLEDGED
MISSING ZERO ADDRESS CHECK	Low	ACKNOWLEDGED
MISSING ZERO VALUE CHECK	Low	ACKNOWLEDGED
REDUNDANT CODE	Low	ACKNOWLEDGED
REDUNDANT INITIALIZER	Low	ACKNOWLEDGED
INCORRECT INITIALIZATION	Low	SOLVED - 10/27/2021
REDUNDANT ARITHMETICAL OPERATIONS	Low	ACKNOWLEDGED
INCORRECT EVENT ARGUMENTS	Low	SOLVED - 11/12/2021
FLOATING PRAGMA	Low	SOLVED - 10/28/2021
MISSING EVENT EMITTING	Low	ACKNOWLEDGED
PRAGMA TOO RECENT	Informational	ACKNOWLEDGED
AMBIGUOUS DESIGN	Informational	ACKNOWLEDGED
STATE VARIABLE VISIBILITY NOT SET	Informational	ACKNOWLEDGED
POSSIBLE MISUSE OF PUBLIC FUNCTIONS	Informational	SOLVED - 11/03/2021
DUPLICATE STATEMENTS	Informational	SOLVED - 10/25/2021

FINDINGS & TECH DETAILS

3.1 (HAL-01) IMPROPER APPLICATION OF PRINCIPLE OF LEAST PRIVILEGE - MEDIUM

Description:

In Kernel.sol during initialization, _owner is assigned to both OWNER_ROLE and MANAGER_ROLE, which violates the principle of least privilege. Also, setting the OWNER_ROLE as role admin for MANAGER_ROLE through _setRoleAdmin(MANAGER_ROLE, OWNER_ROLE); has no benefits.

Code Location:

Listing 1: Kernel.sol (Lines 135,138,141)

```
123 function initialize(
124     address admin_,
125     address owner_,
126     address moduleMap_
127   ) external initializer {
128     __ModuleMapConsumer_init(moduleMap_);
129     __AccessControl_init();
130
131     // make the "admin_" address the default admin role
132     _setupRole(DEFAULT_ADMIN_ROLE, admin_);
133
134     // make the "owner_" address the owner of the system
135     _setupRole(OWNER_ROLE, owner_);
136
137     // give the "owner_" address the manager role, too
138     _setupRole(MANAGER_ROLE, owner_);
139
140     // owners are admins of managers
141     _setRoleAdmin(MANAGER_ROLE, OWNER_ROLE);
142
143     initializationTimestamp = block.timestamp;
144   }
```

Risk Level:

Likelihood - 2
Impact - 4

Recommendation:

It is recommended to have different addresses assigned to both OWNER_ROLE and MANAGER_ROLE and update deployment scripts to reflect changes.

Remediation Plan:

RISK ACCEPTED: 0x_nodes accepts the risk and they will fix the issue in a future release.

3.2 (HAL-02) MISSING RE-ENTRANCY PROTECTION - MEDIUM

Description:

To protect against cross-function reentrancy attacks, it may be necessary to use a mutex. By using this lock, an attacker can no longer exploit the function with a recursive call. OpenZeppelin has it's own mutex implementation called ReentrancyGuard which provides a modifier to any function called "nonReentrant" that guards the function with a mutex against the Reentrancy attacks.

Code Location:

There are many functions within the project that are missing nonReentrant modifier.

Kernel.sol
- deposit
- withdraw
- claimEthRewards
- claimBiosRewards
- claimAllRewards
- enterStrategy
- exitStrategy

Risk Level:

Likelihood - 3
Impact - 3

Recommendation:

It is recommended to follow the checks-effects-interactions pattern and use ReentrancyGuard through the nonReentrant modifier.

Remediation Plan:

SOLVED: 0x_nodes team added the nonReentrant modifier to all required functions.

3.3 (HAL-03) LACK OF PAUSE FUNCTION - LOW

Description:

The project lacks ability to pause contracts. It is advised that in case of unexpected events temporarily disable some important functions to prevent further damage.

Risk Level:

Likelihood - 1
Impact - 4

Remediation:

Consider implementing the pause feature in the smart contracts. It can be achieved by using OpenZepplin's PausableUpgradeable contracts. Also, it is recommended to add a separate role for being responsible for pausing smart contracts when needed.

References:

- https://github.com/OpenZeppelin/openzeppelin-contracts-upgradeable/blob/master/contracts/security/PausableUpgradeable.sol

Remediation Plan:

ACKNOWLEDGED: 0x_nodes acknowledged the issue and they will fix it in a future release.

3.4 (HAL-04) DIVIDE BEFORE MULTIPLY - LOW

Description:

Solidity integer division might truncate. As a result, performing multiplication before division can sometimes avoid loss of precision. In SushiSwapIntegration.sol, there are two instances of multiplication performed on the result of division.

Code Location:

Listing 2: SushiSwapIntegration.sol (Lines 134,135)

```
124     function getPoolEthValue(uint256 poolid) public view returns (
            uint256 poolETHValue) {
125         address token0 = pools[poolid].tokenPair.token0;
126         address token1 = pools[poolid].tokenPair.token1;
127         uint256 lpAmount = pools[poolid].poolInfo.lpToken.
                balanceOf(address(this)) + ISushiSwapMasterChef(
                masterChef).userInfo(poolid, address(this)).amount;
128         uint256 sharePercent = (lpAmount * 10000000000 / pools[
                poolid].poolInfo.lpToken.totalSupply());
129
130         if(sharePercent == 0) {
131             return 0;
132         }
133
134         uint256 token0Amount = sharePercent / 2 * IERC20(token0).
                balanceOf(address(pools[poolid].poolInfo.lpToken)) /
                10000000000;
135         uint256 token1Amount = sharePercent / 2 * IERC20(token1).
                balanceOf(address(pools[poolid].poolInfo.lpToken)) /
                10000000000;
136
137
138
139         return wethAddress == address(token0) ? token0Amount +
                getAmountOut(token1, token0, token1Amount) :
                token1Amount + getAmountOut(token0, token1,
```

```
                token0Amount);
140     }
```

Risk Level:

Likelihood - 2
Impact - 3

Recommendation:

Consider doing multiplication operation before division to prevail precision in the values in non-floating data type. The sample solution is shared in the code snippet below.

Listing 3: SushiSwapIntegration.sol (Lines 134,135)

```
124     function getPoolEthValue(uint256 poolid) public view returns (
            uint256 poolETHValue) {
125         address token0 = pools[poolid].tokenPair.token0;
126         address token1 = pools[poolid].tokenPair.token1;
127         uint256 lpAmount = pools[poolid].poolInfo.lpToken.
                balanceOf(address(this)) + ISushiSwapMasterChef(
                masterChef).userInfo(poolid, address(this)).amount;
128         uint256 sharePercent = (lpAmount * 10000000000 / pools[
                poolid].poolInfo.lpToken.totalSupply());
129
130         if(sharePercent == 0) {
131             return 0;
132         }
133
134         uint256 token0Amount = sharePercent / (2 * IERC20(token0).
                balanceOf(address(pools[poolid].poolInfo.lpToken))) /
                10000000000;
135         uint256 token1Amount = sharePercent / (2 * IERC20(token1).
                balanceOf(address(pools[poolid].poolInfo.lpToken))) /
                10000000000;
136
137
138
```

```
139         return wethAddress == address(token0) ? token0Amount +
                getAmountOut(token1, token0, token1Amount) :
                token1Amount + getAmountOut(token0, token1,
                token0Amount);
140     }
```

Remediation Plan:

SOLVED: 0x_nodes team fixed the code to perform multiplication first.

3.5 (HAL-05) USE OF BLOCK.TIMESTAMP - LOW

Description:

There are many instances of block.timestamp usage within the project. block.timestamp can be influenced by miners to a certain degree, so developers should be aware that this may have some risk if miners collude on time manipulation to influence the price oracles.

Code Location:

Listing 4: Kernel.sol (Lines 143)

```
123 function initialize(
124     address admin_,
125     address owner_,
126     address moduleMap_
127   ) external initializer {
128     __ModuleMapConsumer_init(moduleMap_);
129     __AccessControl_init();
130
131     // make the "admin_" address the default admin role
132     _setupRole(DEFAULT_ADMIN_ROLE, admin_);
133
134     // make the "owner_" address the owner of the system
135     _setupRole(OWNER_ROLE, owner_);
136
137     // give the "owner_" address the manager role, too
138     _setupRole(MANAGER_ROLE, owner_);
139
140     // owners are admins of managers
141     _setRoleAdmin(MANAGER_ROLE, OWNER_ROLE);
142
143     initializationTimestamp = block.timestamp;
144   }
```

Listing 5: Kernel.sol (Lines 561)

```
559 function deploy() external onlyGasAccount {
560     IYieldManager(moduleMap.getModuleAddress(Modules.YieldManager)
            ).deploy();
561     lastDeployTimestamp = block.timestamp;
562     emit Deploy();
563   }
```

Listing 6: Kernel.sol (Lines 569)

```
566 function harvestYield() external onlyGasAccount {
567     IYieldManager(moduleMap.getModuleAddress(Modules.YieldManager)
            )
568       .harvestYield();
569     lastHarvestYieldTimestamp = block.timestamp;
570     emit HarvestYield();
571   }
```

Listing 7: Kernel.sol (Lines 577)

```
574 function processYield() external onlyGasAccount {
575     IYieldManager(moduleMap.getModuleAddress(Modules.YieldManager)
            )
576       .processYield();
577     lastProcessYieldTimestamp = block.timestamp;
578     emit ProcessYield();
579   }
```

Listing 8: Kernel.sol (Lines 586)

```
582 function distributeEth() external onlyGasAccount {
583     IYieldManager(moduleMap.getModuleAddress(Modules.YieldManager)
        )
584       .distributeEth();
585     lastLastDistributeEthTimestamp = lastDistributeEthTimestamp;
586     lastDistributeEthTimestamp = block.timestamp;
587     emit DistributeEth();
588   }
```

Listing 9: Kernel.sol (Lines 594)

```
591 function biosBuyBack() external onlyGasAccount {
592     IYieldManager(moduleMap.getModuleAddress(Modules.YieldManager)
        )
593       .biosBuyBack();
594     lastBiosBuyBackTimestamp = block.timestamp;
595     emit BiosBuyBack();
596   }
```

Listing 10: BiosRewards.sol (Lines 55,58,62,63)

```
50  function notifyRewardAmount(
51    address token,
52    uint256 reward,
53    uint32 duration
54  ) external override onlyController updateReward(token, address
        (0)) {
55    if (block.timestamp >= periodFinish[token]) {
56      rewardRate[token] = reward / duration;
57    } else {
58      uint256 remaining = periodFinish[token] - block.timestamp;
59      uint256 leftover = remaining * rewardRate[token];
60      rewardRate[token] = (reward + leftover) / duration;
61    }
62    lastUpdateTime[token] = block.timestamp;
63    periodFinish[token] = block.timestamp + duration;
64    totalBiosRewards += reward;
65    emit RewardAdded(token, reward, duration);
66  }
```

Listing 11: BiosRewards.sol (Lines 107)

```
101 function lastTimeRewardApplicable(address token)
102   public
103   view
104   override
105   returns (uint256)
106 {
107   return MathUpgradeable.min(block.timestamp, periodFinish[token
          ]);
108 }
```

Listing 12: SushiSwapTrader.sol (Lines 140)

```
125   function swapExactInput(
126     address tokenIn,
127     address tokenOut,
128     address recipient,
129     uint256 amountIn,
130     uint256 amountOutMin
131   ) public override onlyController returns (bool) {
132     require(
133       IERC20MetadataUpgradeable(tokenIn).balanceOf(address(this))
              >= amountIn,
134       "SushiSwapTrader::swapExactInput: Balance is less than trade
              amount"
135     );
136
137     address[] memory path = new address[](2);
138     path[0] = tokenIn;
139     path[1] = tokenOut;
140     uint256 deadline = block.timestamp;
141     ...
142     }
```

- Also, anywhere deadline is used.

Listing 13: UniswapTrader.sol (Lines 306,343)

```
286   function swapExactInput(
287     address tokenIn,
288     address tokenOut,
289     address recipient,
290     uint256 amountIn
291   ) external override onlyController returns (bool tradeSuccess) {
292     IERC20MetadataUpgradeable tokenInErc20 =
            IERC20MetadataUpgradeable(tokenIn);
293
294     if (isMultihopPair[tokenIn][tokenOut]) {
295       Path memory path = getPathFor(tokenIn, tokenOut);
296       IUniswapSwapRouter.ExactInputParams memory params =
              IUniswapSwapRouter
297         .ExactInputParams({
298           path: abi.encodePacked(
299             path.tokenIn,
300             path.firstPoolFee,
301             path.tokenInTokenOut,
302             path.secondPoolFee,
303             path.tokenOut
304           ),
305           recipient: recipient,
306           deadline: block.timestamp,
307           amountIn: amountIn,
308           amountOutMinimum: 0
309         });
310
311       // Executes the swap.
312       try IUniswapSwapRouter(swapRouterAddress).exactInput(params)
              {
313         tradeSuccess = true;
314       } catch {
315         tradeSuccess = false;
316         tokenInErc20.safeTransfer(
317           recipient,
318           tokenInErc20.balanceOf(address(this))
319         );
320       }
321
322       return tradeSuccess;
323     }
324
325     (address token0, address token1) = getTokensSorted(tokenIn,
```

```
            tokenOut);
326
327       require(
328         pools[token0][token1].length > 0,
329         "UniswapTrader::swapExactInput: Pool has not been added"
330       );
331       require(
332         tokenInErc20.balanceOf(address(this)) >= amountIn,
333         "UniswapTrader::swapExactInput: Balance is less than trade
              amount"
334       );
335
336       uint256 amountOutMinimum = getAmountOutMinimum(tokenIn,
            tokenOut, amountIn);
337
338       IUniswapSwapRouter.ExactInputSingleParams memory
            exactInputSingleParams;
339       exactInputSingleParams.tokenIn = tokenIn;
340       exactInputSingleParams.tokenOut = tokenOut;
341       exactInputSingleParams.fee = pools[token0][token1][0].
            feeNumerator;
342       exactInputSingleParams.recipient = recipient;
343       exactInputSingleParams.deadline = block.timestamp;
344       exactInputSingleParams.amountIn = amountIn;
345       exactInputSingleParams.amountOutMinimum = amountOutMinimum;
346       exactInputSingleParams.sqrtPriceLimitX96 = 0;
347       ...
348     }
```

Listing 14: UniswapTrader.sol (Lines 388,422)

```
368   function swapExactOutput(
369     address tokenIn,
370     address tokenOut,
371     address recipient,
372     uint256 amountOut
373   ) external override onlyController returns (bool tradeSuccess) {
374     IERC20MetadataUpgradeable tokenInErc20 =
          IERC20MetadataUpgradeable(tokenIn);
375
376     if (isMultihopPair[tokenIn][tokenOut]) {
377       Path memory path = getPathFor(tokenIn, tokenOut);
378       IUniswapSwapRouter.ExactOutputParams memory params =
            IUniswapSwapRouter
379         .ExactOutputParams({
380           path: abi.encodePacked(
381             path.tokenIn,
382             path.firstPoolFee,
383             path.tokenInTokenOut,
384             path.secondPoolFee,
385             path.tokenOut
386           ),
387           recipient: recipient,
388           deadline: block.timestamp,
389           amountOut: amountOut,
390           amountInMaximum: 0
391         });
392
393       // Executes the swap.
394       try IUniswapSwapRouter(swapRouterAddress).exactOutput(params
            ) {
395         tradeSuccess = true;
396       } catch {
397         tradeSuccess = false;
398         tokenInErc20.safeTransfer(
399           recipient,
400           tokenInErc20.balanceOf(address(this))
401         );
402       }
403
404       return tradeSuccess;
405     }
406     (address token0, address token1) = getTokensSorted(tokenIn,
          tokenOut);
```

```
407     require(
408       pools[token0][token1][0].feeNumerator > 0,
409       "UniswapTrader::swapExactOutput: Pool has not been added"
410     );
411     uint256 amountInMaximum = getAmountInMaximum(tokenIn, tokenOut
            , amountOut);
412     require(
413       tokenInErc20.balanceOf(address(this)) >= amountInMaximum,
414       "UniswapTrader::swapExactOutput: Balance is less than trade
            amount"
415     );
416
417     IUniswapSwapRouter.ExactOutputSingleParams memory
            exactOutputSingleParams;
418     exactOutputSingleParams.tokenIn = tokenIn;
419     exactOutputSingleParams.tokenOut = tokenOut;
420     exactOutputSingleParams.fee = pools[token0][token1][0].
            feeNumerator;
421     exactOutputSingleParams.recipient = recipient;
422     exactOutputSingleParams.deadline = block.timestamp;
423     exactOutputSingleParams.amountOut = amountOut;
424     exactOutputSingleParams.amountInMaximum = amountInMaximum;
425     exactOutputSingleParams.sqrtPriceLimitX96 = 0;
426     ...
427   }
```

Listing 15: DynamicRangeOrdersIntegration.sol (Lines 517)

```
476     function rerangeLiquidityPosition(bytes32 liquidityPositionKey
            )
477     public
478     onlyController
479   {
480     uint256 positionActualBaseStablecoinValue =
            getPositionBaseStablecoinValue(
481       liquidityPositionKey
482     );
483     dynamicRangeOrdersIntegrationDeployer.
            decreaseLiquidityPosition(
484       liquidityPositionKey,
485       positionActualBaseStablecoinValue
486     );
487
488     (
489       uint256 amount0Desired,
490       uint256 amount1Desired
491     ) = getIncreaseLiquidityAmounts(
492         liquidityPositionKey,
493         liquidityPositionKeyIndexes[liquidityPositionKey]
494       );
495
496     if (amount0Desired > 0 || amount1Desired > 0) {
497       (
498         bool success,
499         uint256 liquidityPositionId
500       ) = dynamicRangeOrdersIntegrationDeployer.
              mintLiquidityPosition(
501           liquidityPositions[liquidityPositionKey].token0,
502           liquidityPositions[liquidityPositionKey].token1,
503           liquidityPositions[liquidityPositionKey].feeNumerator,
504           liquidityPositions[liquidityPositionKey].tickLower,
505           liquidityPositions[liquidityPositionKey].tickUpper,
506           amount0Desired,
507           amount1Desired
508         );
509
510       // If the mint succeeded, update the liquidity position data
511       if (success) {
512         liquidityPositions[liquidityPositionKey].minted = true;
513         liquidityPositions[liquidityPositionKey].id =
              liquidityPositionId;
```

```
514         }
515       }
516
517       lastRerangeTimestamp[liquidityPositionKey] = block.timestamp;
518     }
```

Listing 16: DynamicRangeOrdersIntegrationDeployer.sol (Lines 228)

```
184 function mintLiquidityPosition(
185     address token0,
186     address token1,
187     uint24 fee,
188     int24 tickLower,
189     int24 tickUpper,
190     uint256 amount0Desired,
191     uint256 amount1Desired
192   )
193     external
194     override
195     onlyController
196     returns (bool success, uint256 liquidityPositionId)
197   {
198     if (
199       amount0Desired >
200       IERC20MetadataUpgradeable(token0).balanceOf(address(this))
201     ) {
202       amount0Desired = IERC20MetadataUpgradeable(token0).balanceOf
            (
203         address(this)
204       );
205     }
206
207     if (
208       amount1Desired >
209       IERC20MetadataUpgradeable(token1).balanceOf(address(this))
210     ) {
211       amount1Desired = IERC20MetadataUpgradeable(token1).balanceOf
            (
212         address(this)
213       );
214     }
215
216     IUniswapPositionManager.MintParams memory mintParams;
217
218     mintParams.token0 = token0;
219     mintParams.token1 = token1;
220     mintParams.fee = fee;
221     mintParams.tickLower = tickLower;
222     mintParams.tickUpper = tickUpper;
223     mintParams.amount0Desired = amount0Desired;
224     mintParams.amount1Desired = amount1Desired;
```

```
225     mintParams.amount0Min = 0;
226     mintParams.amount1Min = 0;
227     mintParams.recipient = address(this);
228     mintParams.deadline = block.timestamp;
229     ...
230     }
```

Listing 17: SushiSwapIntegration.sol (Lines 211)

```
169 function _deploy(uint256 amount, uint256 pid) internal {
170         ISushiSwapIntegration.InnerPool memory innerPoolInfo =
                getPoolInfo(pid);
171         require(innerPoolInfo.added, "SushiSwapIntegration::
                deposit to sushiSwap yield farm: pool is not configured
                ");
172
173         (uint256 priceWithMultiplier, uint256
                decimalsSubtractionExponent) = getPriceWithMultiplier(
                getTokenToAdd(innerPoolInfo.tokenPair.token0,
                innerPoolInfo.tokenPair.token1), address(innerPoolInfo.
                poolInfo.lpToken), innerPoolInfo.tokenPair.token0);
174         (uint256 amountTokenDesired, uint256 amountTokenMin,
                uint256 amountWeiMin) =
                calculateAmountOfTokenToAddLiquidityETH(amount / 2,
                priceWithMultiplier);
175         uint256 tokensReceived;
176
177         if(IERC20MetadataUpgradeable(IIntegrationMap(moduleMap.
                getModuleAddress(Modules.IntegrationMap)).
                getWethTokenAddress()).balanceOf(address(this)) <
                amount) {
178             return;
179         }
180         IWeth9(IIntegrationMap(moduleMap.getModuleAddress(Modules.
                IntegrationMap)).getWethTokenAddress()).withdraw(amount
                );
181
182         if (IERC20MetadataUpgradeable(getTokenToAdd(innerPoolInfo.
                tokenPair.token0, innerPoolInfo.tokenPair.token1)).
                balanceOf(moduleMap.getModuleAddress(Modules.Kernel)) <
                 amountTokenDesired / decimalsSubtractionExponent) {
183
184             uint[] memory amounts = swapExactETHForTokens(
185                 amount / 2,
186                 ((amountTokenDesired - (amountTokenDesired * 1 /
                        100)) / decimalsSubtractionExponent),
187                 wethAddress,
188                 getTokenToAdd(innerPoolInfo.tokenPair.token0,
                        innerPoolInfo.tokenPair.token1),
189                 address(this)
190             );
191
```

```
192             tokensReceived = amounts[1];
193
194             emit TokensReceived(tokensReceived);
195
196             (priceWithMultiplier,) = getPriceWithMultiplier(
                    getTokenToAdd(innerPoolInfo.tokenPair.token0,
                    innerPoolInfo.tokenPair.token1), address(
                    innerPoolInfo.poolInfo.lpToken), innerPoolInfo.
                    tokenPair.token0);
197             (amountTokenDesired, amountTokenMin, amountWeiMin) =
                    calculateAmountOfTokenToAddLiquidityETH(amount / 2,
                     priceWithMultiplier);
198         }
199
200         IERC20MetadataUpgradeable(getTokenToAdd(innerPoolInfo.
                tokenPair.token0, innerPoolInfo.tokenPair.token1)).
                safeApprove(swapRouterAddress, 0);
201         IERC20MetadataUpgradeable(getTokenToAdd(innerPoolInfo.
                tokenPair.token0, innerPoolInfo.tokenPair.token1)).
                safeApprove(swapRouterAddress,
                IERC20MetadataUpgradeable(getTokenToAdd(innerPoolInfo.
                tokenPair.token0, innerPoolInfo.tokenPair.token1)).
                allowance(address(this), swapRouterAddress) + (
                tokensReceived == 0 ? amountTokenDesired /
                decimalsSubtractionExponent : tokensReceived));
202
203         uint256 diff = amountTokenDesired /
                decimalsSubtractionExponent - (tokensReceived == 0 ?
                amountTokenDesired / decimalsSubtractionExponent :
                tokensReceived);
204
205         (,,uint256 liquidity) = ISushiSwapRouter(swapRouterAddress
                ).addLiquidityETH{value: amount / 2}(
206             getTokenToAdd(innerPoolInfo.tokenPair.token0,
                    innerPoolInfo.tokenPair.token1),
207             (amountTokenDesired / decimalsSubtractionExponent) -
                    diff,
208             (amountTokenMin / decimalsSubtractionExponent),
209             amountWeiMin,
210             address(this),
211             block.timestamp + 360
212         );
213         ...}
```

Listing 18: SushiSwapIntegration.sol (Lines 353)

```
333     function _withdraw(uint256 poolid, uint amountPercent)
            internal returns(uint256) {
334         uint256 liquidity = ISushiSwapMasterChef(masterChef).
                userInfo(poolid, address(this)).amount * amountPercent
                / 10000;
335
336         ISushiSwapIntegration.InnerPool memory innerPoolInfo =
                getPoolInfo(poolid);
337
338         ISushiSwapMasterChef(masterChef).withdraw(poolid,
                liquidity);
339         IERC20MetadataUpgradeable(address(innerPoolInfo.poolInfo.
                lpToken)).safeApprove(swapRouterAddress, liquidity);
340
341         address token = getTokenToAdd(innerPoolInfo.tokenPair.
                token0, innerPoolInfo.tokenPair.token1);
342
343         (uint256 priceWithMultiplier, uint256
                decimalsSubtractionExponent) = getPriceWithMultiplier(
                token, address(innerPoolInfo.poolInfo.lpToken),
                innerPoolInfo.tokenPair.token0);
344
345         (, uint256 amountTokenMin, uint256 amountWeiMin) =
                calculateAmountOfTokenToAddLiquidityETH(poolBalances[
                poolid] / 2, priceWithMultiplier);
346
347         (uint amountToken, ) = ISushiSwapRouter(swapRouterAddress)
                .removeLiquidityETH(
348             token,
349             liquidity,
350             amountTokenMin / decimalsSubtractionExponent,
351             amountWeiMin,
352             address(this),
353             block.timestamp + 360
354         );
355         ...}
```

Listing 19: SushiSwapIntegration.sol (Lines 372)

```
362     function swapExactETHForTokens(
363         uint256 amountWei,
364         uint256 amountOutMin,
365         address tokenIn,
366         address tokenOut,
367         address to
368     ) internal returns (uint[] memory) {
369         address[] memory path = new address[](2);
370         path[0] = tokenIn;
371         path[1] = tokenOut;
372         uint256 deadline = block.timestamp;
373 
374         return ISushiSwapRouter(swapRouterAddress).
                swapExactETHForTokens{value: amountWei}(amountOutMin,
                path, to, deadline);
375     }
```

Listing 20: SushiSwapIntegration.sol (Lines 391)

```
381     function swapExactInput(
382         address tokenIn,
383         address tokenOut,
384         address recipient,
385         uint256 amountIn
386     ) public override onlyController returns (uint[] memory) {
387         uint256 amountOutMin = getAmountOutMinimum(tokenIn,
                tokenOut, amountIn);
388         address[] memory path = new address[](2);
389         path[0] = tokenIn;
390         path[1] = tokenOut;
391         uint256 deadline = block.timestamp;
392 
393         IERC20MetadataUpgradeable(tokenIn).approve(
                swapRouterAddress, amountIn);
394 
395         return ISushiSwapRouter(swapRouterAddress).
                swapExactTokensForTokens(amountIn, amountOutMin, path,
                recipient, deadline);
396     }
```

Listing 21: UniswapIntegrationDeployer.sol (Lines 226)

```
182   function mintLiquidityPosition(
183     address token0,
184     address token1,
185     uint24 fee,
186     int24 tickLower,
187     int24 tickUpper,
188     uint256 amount0Desired,
189     uint256 amount1Desired
190   )
191     external
192     override
193     onlyController
194     returns (bool success, uint256 liquidityPositionId)
195   {
196     if (
197       amount0Desired >
198       IERC20MetadataUpgradeable(token0).balanceOf(address(this))
199     ) {
200       amount0Desired = IERC20MetadataUpgradeable(token0).balanceOf
          (
201         address(this)
202       );
203     }
204
205     if (
206       amount1Desired >
207       IERC20MetadataUpgradeable(token1).balanceOf(address(this))
208     ) {
209       amount1Desired = IERC20MetadataUpgradeable(token1).balanceOf
          (
210         address(this)
211       );
212     }
213
214     IUniswapPositionManager.MintParams memory mintParams;
215
216     mintParams.token0 = token0;
217     mintParams.token1 = token1;
218     mintParams.fee = fee;
219     mintParams.tickLower = tickLower;
220     mintParams.tickUpper = tickUpper;
221     mintParams.amount0Desired = amount0Desired;
222     mintParams.amount1Desired = amount1Desired;
```

```
223     mintParams.amount0Min = 0;
224     mintParams.amount1Min = 0;
225     mintParams.recipient = address(this);
226     mintParams.deadline = block.timestamp;
227     ...
228     }
```

Listing 22: UniswapIntegrationDeployer.sol (Lines 277)

```
245   function increaseLiquidityPosition(
246     uint256 liquidityPositionId,
247     address token0,
248     address token1,
249     uint256 amount0Desired,
250     uint256 amount1Desired
251   ) external override onlyController returns (bool success) {
252     if (
253       amount0Desired >
254       IERC20MetadataUpgradeable(token0).balanceOf(address(this))
255     ) {
256       amount0Desired = IERC20MetadataUpgradeable(token0).balanceOf
          (
257         address(this)
258       );
259     }
260     if (
261       amount1Desired >
262       IERC20MetadataUpgradeable(token1).balanceOf(address(this))
263     ) {
264       amount1Desired = IERC20MetadataUpgradeable(token1).balanceOf
          (
265         address(this)
266       );
267     }
268
269     IUniswapPositionManager.IncreaseLiquidityParams
270       memory increaseLiquidityParams;
271
272     increaseLiquidityParams.tokenId = liquidityPositionId;
273     increaseLiquidityParams.amount0Desired = amount0Desired;
274     increaseLiquidityParams.amount1Desired = amount1Desired;
275     increaseLiquidityParams.amount0Min = 0;
276     increaseLiquidityParams.amount1Min = 0;
277     increaseLiquidityParams.deadline = block.timestamp;
278     ...
279     }
```

Listing 23: UniswapIntegrationDeployer.sol (Lines 450)

```
404  function decreaseLiquidityPosition(
405    uint256 liquidityPositionIndex,
406    uint256 baseStablecoinValue
407   ) public override onlyController returns (bool success) {
408    (, , , , , , uint256 liquidityPositionId, ) =
           uniswapIntegration
409      .getLiquidityPosition(liquidityPositionIndex);
410    (
411      ,
412      ,
413      ,
414      ,
415      ,
416      ,
417      ,
418      uint128 currentLiquidity,
419      ,
420      ,
421      uint128 tokensOwed0Before,
422      uint128 tokensOwed1Before
423    ) = IUniswapPositionManager(positionManagerAddress).positions(
424        liquidityPositionId
425      );
426
427    if (
428      uniswapIntegration.getPositionBaseStablecoinValue(
429        liquidityPositionIndex
430      ) > 0
431    ) {
432      uint128 reduceLiquidityAmount = uint128(
433        ((baseStablecoinValue) * currentLiquidity) /
434          uniswapIntegration.getPositionBaseStablecoinValue(
435            liquidityPositionIndex
436          )
437      );
438
439      if (reduceLiquidityAmount > currentLiquidity) {
440        reduceLiquidityAmount = currentLiquidity;
441      }
442
443      IUniswapPositionManager.DecreaseLiquidityParams
444        memory decreaseLiquidityParams;
445
```

```
446         decreaseLiquidityParams.tokenId = liquidityPositionId;
447         decreaseLiquidityParams.liquidity = reduceLiquidityAmount;
448         decreaseLiquidityParams.amount0Min = 0;
449         decreaseLiquidityParams.amount1Min = 0;
450         decreaseLiquidityParams.deadline = block.timestamp;
451         ...
452         }
```

Risk Level:

Likelihood - 2
Impact - 3

Recommendation:

Use block.number instead of block.timestamp or now to reduce the risk of Maximal Extractable Value (MEV) attacks. Check if the timescale of the project occurs across years, days and months rather than seconds. If possible, it is recommended to use Oracles.

Remediation Plan:

NOT APPLICABLE: 0x_nodes considers the usage of block.timestamp safe, since the primary use of timestamps in the system is used by the offchain processor when scheduling updates and fund deployment. The functions that use these timestamps are secured via openzeppelin Ownable, and cannot be exploited by a miner. They are also intended to function on a scale of hours, not seconds.

3.6 (HAL-06) EXTERNAL FUNCTION CALLS WITHIN LOOP - LOW

Description:

There are many instances within contracts that use external function calls within a loop. Calls inside a loop might lead to a denial-of-service attack.

Code Location:

Listing 24: BiosRewards.sol (Lines)

```
1 BiosRewards.getUserBiosRewards(address) (contracts/core/
    BiosRewards.sol#141-160) has external calls inside a loop:
    tokenId < integrationMap.getTokenAddressesLength() (contracts/
    core/BiosRewards.sol#154-156)
2 BiosRewards.getUserBiosRewards(address) (contracts/core/
    BiosRewards.sol#141-160) has external calls inside a loop:
    userBiosRewards += earned(integrationMap.getTokenAddress(
    tokenId),account) (contracts/core/BiosRewards.sol#156-160)
3 EtherRewards.claimEthRewards(address) (contracts/core/EtherRewards
    .sol#76-90) has external calls inside a loop: token =
    IIntegrationMap(integrationMap).getTokenAddress(tokenId) (
    contracts/core/EtherRewards.sol#87)
4 EtherRewards.getUserEthRewards(address) (contracts/core/
    EtherRewards.sol#131-145) has external calls inside a loop:
    token = IIntegrationMap(integrationMap).getTokenAddress(tokenId
    ) (contracts/core/EtherRewards.sol#142)
5 StrategyMap.updateIntegrations(uint256,IStrategyMap.
    WeightedIntegration[]) (contracts/core/StrategyMap.sol#121-193)
     has external calls inside a loop: token = integrationMap.
    getTokenAddress(i_scope_1) (contracts/core/StrategyMap.sol#164)
6 StrategyMap.deleteStrategy(uint256) (contracts/core/StrategyMap.
    sol#195-226) has external calls inside a loop: require(bool,
    string)(getStrategyTokenBalance(id,integrationMap.
    getTokenAddress(i)) == 0,Strategy in use) (contracts/core/
    StrategyMap.sol#204-207)
7 StrategyMap.enterStrategy(uint256,address,address[],uint256[]) (
    contracts/core/StrategyMap.sol#276-312) has external calls
    inside a loop: require(bool,string)(integrationMap.
```

```
    getTokenAcceptingDeposits(tokens[i]),Token unavailable) (
    contracts/core/StrategyMap.sol#294-297)
8 StrategyMap.enterStrategy(uint256,address,address[],uint256[]) (
    contracts/core/StrategyMap.sol#276-312) has external calls
    inside a loop: require(bool,string)(userPositions.
    userTokenBalance(tokens[i],user) >= amounts[i],User lacks funds
    ) (contracts/core/StrategyMap.sol#300-303)
9 StrategyMap.exitStrategy(uint256,address,address[],uint256[]) (
    contracts/core/StrategyMap.sol#314-345) has external calls
    inside a loop: require(bool,string)(integrationMap.
    getTokenAcceptingWithdrawals(tokens[i]),Token unavailable) (
    contracts/core/StrategyMap.sol#331-334)
10 UserPositions.deposit(address,address[],uint256[],uint256) (
    contracts/core/UserPositions.sol#98-166) has external calls
    inside a loop: require(bool,string)(integrationMap.
    getTokenAcceptingDeposits(tokens[tokenId]),UserPositions::
    deposit: This token is not accepting deposits) (contracts/core/
    UserPositions.sol#110-113)
11 UserPositions.deposit(address,address[],uint256[],uint256) (
    contracts/core/UserPositions.sol#98-166) has external calls
    inside a loop: beforeBalance = erc20.balanceOf(moduleMap.
    getModuleAddress(Modules.Kernel)) (contracts/core/UserPositions
    .sol#125-127)
12 UserPositions.deposit(address,address[],uint256[],uint256) (
    contracts/core/UserPositions.sol#98-166) has external calls
    inside a loop: erc20.safeTransferFrom(depositor,moduleMap.
    getModuleAddress(Modules.Kernel),amounts[tokenId]) (contracts/
    core/UserPositions.sol#130-134)
13 UserPositions.deposit(address,address[],uint256[],uint256) (
    contracts/core/UserPositions.sol#98-166) has external calls
    inside a loop: afterBalance = erc20.balanceOf(moduleMap.
    getModuleAddress(Modules.Kernel)) (contracts/core/UserPositions
    .sol#137-139)
14 UserPositions.deposit(address,address[],uint256[],uint256) (
    contracts/core/UserPositions.sol#98-166) has external calls
    inside a loop: IBiosRewards(moduleMap.getModuleAddress(Modules.
    BiosRewards)).increaseRewards(tokens[tokenId],depositor,
    actualAmount) (contracts/core/UserPositions.sol#143-144)
15 UserPositions.deposit(address,address[],uint256[],uint256) (
    contracts/core/UserPositions.sol#98-166) has external calls
    inside a loop: IEtherRewards(moduleMap.getModuleAddress(Modules
    .EtherRewards)).updateUserRewards(tokens[tokenId],depositor) (
    contracts/core/UserPositions.sol#145-146)
```

```
16 UserPositions._withdraw(address,address[],uint256[],bool) (
    contracts/core/UserPositions.sol#233-304) has external calls
    inside a loop: require(bool,string)(integrationMap.
    getTokenAcceptingWithdrawals(tokens[tokenId]),UserPositions::
    _withdraw: This token is not accepting withdrawals) (contracts/
    core/UserPositions.sol#254-257)
17 UserPositions._withdraw(address,address[],uint256[],bool) (
    contracts/core/UserPositions.sol#233-304) has external calls
    inside a loop: IERC20MetadataUpgradeable(tokens[tokenId]).
    balanceOf(moduleMap.getModuleAddress(Modules.Kernel)) < amounts
    [tokenId] (contracts/core/UserPositions.sol#264-266)
18 UserPositions._withdraw(address,address[],uint256[],bool) (
    contracts/core/UserPositions.sol#233-304) has external calls
    inside a loop: IERC20MetadataUpgradeable(tokens[tokenId]).
    balanceOf(moduleMap.getModuleAddress(Modules.Kernel)) < amounts
    [tokenId] (contracts/core/UserPositions.sol#272-274)
19 UserPositions._withdraw(address,address[],uint256[],bool) (
    contracts/core/UserPositions.sol#233-304) has external calls
    inside a loop: amounts[tokenId] = IERC20MetadataUpgradeable(
    tokens[tokenId]).balanceOf(moduleMap.getModuleAddress(Modules.
    Kernel)) (contracts/core/UserPositions.sol#277-278)
20 UserPositions._withdraw(address,address[],uint256[],bool) (
    contracts/core/UserPositions.sol#233-304) has external calls
    inside a loop: IBiosRewards(moduleMap.getModuleAddress(Modules.
    BiosRewards)).decreaseRewards(tokens[tokenId],recipient,amounts
    [tokenId]) (contracts/core/UserPositions.sol#294-295)
21 UserPositions._withdraw(address,address[],uint256[],bool) (
    contracts/core/UserPositions.sol#233-304) has external calls
    inside a loop: IEtherRewards(moduleMap.getModuleAddress(Modules
    .EtherRewards)).updateUserRewards(tokens[tokenId],recipient) (
    contracts/core/UserPositions.sol#297-298)
22 UserPositions._withdraw(address,address[],uint256[],bool) (
    contracts/core/UserPositions.sol#233-304) has external calls
    inside a loop: IERC20MetadataUpgradeable(tokens[tokenId]).
    safeTransferFrom(moduleMap.getModuleAddress(Modules.Kernel),
    recipient,amounts[tokenId]) (contracts/core/UserPositions.sol
    #286-290)
23 UserPositions.closePositionsForWithdrawal(address,uint256) (
    contracts/core/UserPositions.sol#308-351) has external calls
    inside a loop: integrationId < integrationMap.
    getIntegrationAddressesLength() (contracts/core/UserPositions.
    sol#321)
24 UserPositions.closePositionsForWithdrawal(address,uint256) (
    contracts/core/UserPositions.sol#308-351) has external calls
```

```
      inside a loop: integrationAddress = integrationMap.
      getIntegrationAddress(integrationId) (contracts/core/
      UserPositions.sol#324-326)
25 UserPositions.closePositionsForWithdrawal(address,uint256) (
      contracts/core/UserPositions.sol#308-351) has external calls
      inside a loop: desiredWithdrawAmount = (amount * strategyMap.
      getIntegrationWeight(integrationAddress)) /
      integrationWeightSum (contracts/core/UserPositions.sol#327-329)
26 UserPositions.closePositionsForWithdrawal(address,uint256) (
      contracts/core/UserPositions.sol#308-351) has external calls
      inside a loop: desiredWithdrawAmount > IIntegration(
      integrationAddress).getBalance(token) (contracts/core/
      UserPositions.sol#332-333)
27 UserPositions.closePositionsForWithdrawal(address,uint256) (
      contracts/core/UserPositions.sol#308-351) has external calls
      inside a loop: desiredWithdrawAmount = IIntegration(
      integrationAddress).getBalance(token) (contracts/core/
      UserPositions.sol#335-337)
28 UserPositions.closePositionsForWithdrawal(address,uint256) (
      contracts/core/UserPositions.sol#308-351) has external calls
      inside a loop: IIntegration(integrationAddress).withdraw(token,
      desiredWithdrawAmount) (contracts/core/UserPositions.sol#340)
29 UserPositions.fullyClosePositionsForWithdrawal(address,uint256) (
      contracts/core/UserPositions.sol#355-386) has external calls
      inside a loop: integration = IIntegration(integrationMap.
      getIntegrationAddress(integrationId)) (contracts/core/
      UserPositions.sol#365-367)
30 UserPositions.fullyClosePositionsForWithdrawal(address,uint256) (
      contracts/core/UserPositions.sol#355-386) has external calls
      inside a loop: integration.withdraw(token,integration.
      getBalance(token)) (contracts/core/UserPositions.sol#370)
31 UserPositions.fullyClosePositionsForWithdrawal(address,uint256) (
      contracts/core/UserPositions.sol#355-386) has external calls
      inside a loop: integrationId == integrationMap.
      getIntegrationAddressesLength() - 1 ||
      IERC20MetadataUpgradeable(token).balanceOf(moduleMap.
      getModuleAddress(Modules.Kernel)) >= amount (contracts/core/
      UserPositions.sol#373-377)
32 UserPositions.transferToStrategy(address,address[],uint256[]) (
      contracts/core/UserPositions.sol#394-428) has external calls
      inside a loop: IBiosRewards(moduleMap.getModuleAddress(Modules.
      BiosRewards)).decreaseRewards(tokens[tokenId],recipient,amounts
      [tokenId]) (contracts/core/UserPositions.sol#418-419)
```

```
33 UserPositions.transferToStrategy(address,address[],uint256[]) (
      contracts/core/UserPositions.sol#394-428) has external calls
      inside a loop: IEtherRewards(moduleMap.getModuleAddress(Modules
      .EtherRewards)).updateUserRewards(tokens[tokenId],recipient) (
      contracts/core/UserPositions.sol#421-422)
34 UserPositions.transferFromStrategy(address,address[],uint256[]) (
      contracts/core/UserPositions.sol#436-465) has external calls
      inside a loop: require(bool,string)(integrationMap.
      getTokenAcceptingDeposits(tokens[tokenId]),UserPositions::
      deposit: This token is not accepting deposits) (contracts/core/
      UserPositions.sol#446-449)
35 UserPositions.transferFromStrategy(address,address[],uint256[]) (
      contracts/core/UserPositions.sol#436-465) has external calls
      inside a loop: IBiosRewards(moduleMap.getModuleAddress(Modules.
      BiosRewards)).increaseRewards(tokens[tokenId],recipient,amounts
      [tokenId]) (contracts/core/UserPositions.sol#456-457)
36 UserPositions.transferFromStrategy(address,address[],uint256[]) (
      contracts/core/UserPositions.sol#436-465) has external calls
      inside a loop: IEtherRewards(moduleMap.getModuleAddress(Modules
      .EtherRewards)).updateUserRewards(tokens[tokenId],recipient) (
      contracts/core/UserPositions.sol#458-459)
37 UserPositions._increaseBiosRewards() (contracts/core/UserPositions
      .sol#473-506) has external calls inside a loop: token =
      integrationMap.getTokenAddress(tokenId) (contracts/core/
      UserPositions.sol#498)
38 UserPositions._increaseBiosRewards() (contracts/core/UserPositions
      .sol#473-506) has external calls inside a loop:
      tokenBiosRewardWeight = integrationMap.getTokenBiosRewardWeight
      (token) (contracts/core/UserPositions.sol#499-501)
39 UserPositions._claimBiosRewards(address) (contracts/core/
      UserPositions.sol#561-590) has external calls inside a loop:
      token = integrationMap.getTokenAddress(tokenId) (contracts/core
      /UserPositions.sol#575)
40 UserPositions._claimBiosRewards(address) (contracts/core/
      UserPositions.sol#561-590) has external calls inside a loop:
      biosRewards.earned(token,recipient) > 0 (contracts/core/
      UserPositions.sol#577)
41 UserPositions._claimBiosRewards(address) (contracts/core/
      UserPositions.sol#561-590) has external calls inside a loop:
      biosClaimed += IBiosRewards(moduleMap.getModuleAddress(Modules.
      BiosRewards)).claimReward(token,recipient) (contracts/core/
      UserPositions.sol#578-580)
42 YieldManager._rebalance() (contracts/core/YieldManager.sol
      #140-204) has external calls inside a loop: integration =
```

```
      integrationMap.getIntegrationAddress(i) (contracts/core/
      YieldManager.sol#152)
43 YieldManager._rebalance() (contracts/core/YieldManager.sol
      #140-204) has external calls inside a loop: token =
      integrationMap.getTokenAddress(j) (contracts/core/YieldManager.
      sol#154)
44 YieldManager._rebalance() (contracts/core/YieldManager.sol
      #140-204) has external calls inside a loop: numerator =
      integrationMap.getTokenReserveRatioNumerator(token) (contracts/
      core/YieldManager.sol#155)
45 YieldManager._rebalance() (contracts/core/YieldManager.sol
      #140-204) has external calls inside a loop: grossAmountInvested
       = strategyMap.getExpectedBalance(integration,token) (contracts
      /core/YieldManager.sol#157-160)
46 YieldManager._rebalance() (contracts/core/YieldManager.sol
      #140-204) has external calls inside a loop: actualBalance =
      IIntegration(integration).getBalance(token) (contracts/core/
      YieldManager.sol#165)
47 YieldManager._rebalance() (contracts/core/YieldManager.sol
      #140-204) has external calls inside a loop:
      IERC20MetadataUpgradeable(token).balanceOf(moduleMap.
      getModuleAddress(Modules.Kernel)) >= shortage (contracts/core/
      YieldManager.sol#172-174)
48 YieldManager._rebalance() (contracts/core/YieldManager.sol
      #140-204) has external calls inside a loop: balanceBefore =
      IERC20MetadataUpgradeable(token).balanceOf(integration) (
      contracts/core/YieldManager.sol#176-178)
49 YieldManager._rebalance() (contracts/core/YieldManager.sol
      #140-204) has external calls inside a loop:
      IERC20MetadataUpgradeable(token).safeTransferFrom(moduleMap.
      getModuleAddress(Modules.Kernel),integration,shortage) (
      contracts/core/YieldManager.sol#179-183)
50 YieldManager._rebalance() (contracts/core/YieldManager.sol
      #140-204) has external calls inside a loop: balanceAfter =
      IERC20MetadataUpgradeable(token).balanceOf(integration) (
      contracts/core/YieldManager.sol#184-186)
51 YieldManager._rebalance() (contracts/core/YieldManager.sol
      #140-204) has external calls inside a loop: IIntegration(
      integration).deposit(token,balanceAfter - balanceBefore) (
      contracts/core/YieldManager.sol#188-191)
52 YieldManager._rebalance() (contracts/core/YieldManager.sol
      #140-204) has external calls inside a loop: IIntegration(
      integration).withdraw(token,actualBalance - desiredBalance) (
      contracts/core/YieldManager.sol#196-199)
```

```
53 YieldManager.harvestYield() (contracts/core/YieldManager.sol
      #216-265) has external calls inside a loop: IIntegration(
      integrationMap.getIntegrationAddress(integrationId)).
      harvestYield() (contracts/core/YieldManager.sol#228-229)
54 YieldManager.harvestYield() (contracts/core/YieldManager.sol
      #216-265) has external calls inside a loop: token =
      IERC20MetadataUpgradeable(integrationMap.getTokenAddress(
      tokenId)) (contracts/core/YieldManager.sol#233-235)
55 YieldManager.harvestYield() (contracts/core/YieldManager.sol
      #216-265) has external calls inside a loop:
      harvestedTokenAmount = token.balanceOf(address(this)) (
      contracts/core/YieldManager.sol#242)
56 YieldManager.harvestYield() (contracts/core/YieldManager.sol
      #216-265) has external calls inside a loop: token.safeTransfer(
      moduleMap.getModuleAddress(Modules.Kernel),tokenDesiredReserve
      - tokenActualReserve) (contracts/core/YieldManager.sol#252-255)
57 YieldManager.harvestYield() (contracts/core/YieldManager.sol
      #216-265) has external calls inside a loop: token.safeTransfer(
      moduleMap.getModuleAddress(Modules.Kernel),token.balanceOf(
      address(this))) (contracts/core/YieldManager.sol#258-261)
58 YieldManager.processYield() (contracts/core/YieldManager.sol
      #268-315) has external calls inside a loop: token =
      IERC20MetadataUpgradeable(integrationMap.getTokenAddress(
      tokenId)) (contracts/core/YieldManager.sol#278-280)
59 YieldManager.processYield() (contracts/core/YieldManager.sol
      #268-315) has external calls inside a loop: token.balanceOf(
      address(this)) > 0 (contracts/core/YieldManager.sol#282)
60 YieldManager.processYield() (contracts/core/YieldManager.sol
      #268-315) has external calls inside a loop: wethBalanceBefore =
       weth.balanceOf(address(this)) (contracts/core/YieldManager.sol
      #287)
61 YieldManager.processYield() (contracts/core/YieldManager.sol
      #268-315) has external calls inside a loop: token.safeTransfer(
      moduleMap.getModuleAddress(Modules.UniswapTrader),token.
      balanceOf(address(this))) (contracts/core/YieldManager.sol
      #290-293)
62 YieldManager.processYield() (contracts/core/YieldManager.sol
      #268-315) has external calls inside a loop: IUniswapTrader(
      moduleMap.getModuleAddress(Modules.UniswapTrader)).
      swapExactInput(address(token),address(weth),address(this),token
      .balanceOf(moduleMap.getModuleAddress(Modules.UniswapTrader)))
      (contracts/core/YieldManager.sol#295-301)
63 YieldManager.processYield() (contracts/core/YieldManager.sol
      #268-315) has external calls inside a loop: wethReceived = weth
```

```
      .balanceOf(address(this)) - wethBalanceBefore (contracts/core/
      YieldManager.sol#303)
64 YieldManager.processYield() (contracts/core/YieldManager.sol
      #268-315) has external calls inside a loop: wethReceived = weth
      .balanceOf(address(this)) - getProcessedWethByTokenSum() (
      contracts/core/YieldManager.sol#306-308)
65 YieldManager.ethToRewards(uint256) (contracts/core/YieldManager.
      sol#416-452) has external calls inside a loop: tokenAddress =
      integrationMap.getTokenAddress(tokenId) (contracts/core/
      YieldManager.sol#430)
66 YieldManager.ethToRewards(uint256) (contracts/core/YieldManager.
      sol#416-452) has external calls inside a loop: IEtherRewards(
      moduleMap.getModuleAddress(Modules.EtherRewards)).
      increaseEthRewards(tokenAddress,(ethRewardsAmount *
      processedWethByToken[tokenAddress]) / processedWethByTokenSum)
      (contracts/core/YieldManager.sol#433-438)
67 YieldManager.getProcessedWethSum() (contracts/core/YieldManager.
      sol#572-588) has external calls inside a loop: tokenAddress =
      IIntegrationMap(moduleMap.getModuleAddress(Modules.
      IntegrationMap)).getTokenAddress(tokenId) (contracts/core/
      YieldManager.sol#583-585)
68 YieldManager.getProcessedWethByTokenSum() (contracts/core/
      YieldManager.sol#602-618) has external calls inside a loop:
      processedWethByTokenSum += processedWethByToken[integrationMap.
      getTokenAddress(tokenId)] (contracts/core/YieldManager.sol
      #614-616)
69 YieldManager.getTokenTotalIntegrationBalance(address) (contracts/
      core/YieldManager.sol#622-642) has external calls inside a loop
      : tokenTotalIntegrationBalance += IIntegration(integrationMap.
      getIntegrationAddress(integrationId)).getBalance(tokenAddress)
      (contracts/core/YieldManager.sol#638-640)
70 AaveIntegration.deploy() (contracts/yield-integrations/
      AaveIntegration.sol#88-115) has external calls inside a loop:
      token = IERC20MetadataUpgradeable(integrationMap.
      getTokenAddress(tokenId)) (contracts/yield-integrations/
      AaveIntegration.sol#95-97)
71 AaveIntegration.deploy() (contracts/yield-integrations/
      AaveIntegration.sol#88-115) has external calls inside a loop:
      tokenAmount = token.balanceOf(address(this)) (contracts/yield-
      integrations/AaveIntegration.sol#98)
72 AaveIntegration.deploy() (contracts/yield-integrations/
      AaveIntegration.sol#88-115) has external calls inside a loop:
      token.allowance(address(this),lendingPoolAddress) == 0 (
      contracts/yield-integrations/AaveIntegration.sol#100)
```

```
73 AaveIntegration.deploy() (contracts/yield-integrations/
      AaveIntegration.sol#88-115) has external calls inside a loop:
      IAaveLendingPool(lendingPoolAddress).deposit(address(token),
      tokenAmount,address(this),0) (contracts/yield-integrations/
      AaveIntegration.sol#105-112)
74 AaveIntegration.harvestYield() (contracts/yield-integrations/
      AaveIntegration.sol#118-141) has external calls inside a loop:
      tokenAddress = integrationMap.getTokenAddress(tokenId) (
      contracts/yield-integrations/AaveIntegration.sol#125)
75 AaveIntegration.harvestYield() (contracts/yield-integrations/
      AaveIntegration.sol#118-141) has external calls inside a loop:
      aTokenBalance = IERC20MetadataUpgradeable(aTokenAddress).
      balanceOf(address(this)) (contracts/yield-integrations/
      AaveIntegration.sol#128-129)
76 AaveIntegration.harvestYield() (contracts/yield-integrations/
      AaveIntegration.sol#118-141) has external calls inside a loop:
      IAaveLendingPool(lendingPoolAddress).withdraw(tokenAddress,
      aTokenBalance - balances[tokenAddress],address(moduleMap.
      getModuleAddress(Modules.YieldManager))) (contracts/yield-
      integrations/AaveIntegration.sol#131-137)
77 DynamicRangeOrdersIntegration._deploy() (contracts/yield-
      integrations/DynamicRangeOrdersIntegration.sol#269-337) has
      external calls inside a loop: token0Id = integrationMap.
      getTokenId(liquidityPositions[liquidityPositionKeys[
      liquidityPositionIndex]].token0) (contracts/yield-integrations/
      DynamicRangeOrdersIntegration.sol#307-309)
78 DynamicRangeOrdersIntegration._deploy() (contracts/yield-
      integrations/DynamicRangeOrdersIntegration.sol#269-337) has
      external calls inside a loop: token1Id = integrationMap.
      getTokenId(liquidityPositions[liquidityPositionKeys[
      liquidityPositionIndex]].token1) (contracts/yield-integrations/
      DynamicRangeOrdersIntegration.sol#310-312)
79 DynamicRangeOrdersIntegration.
      transferTokensTodynamicRangeOrdersIntegrationDeployer() (
      contracts/yield-integrations/DynamicRangeOrdersIntegration.sol
      #365-392) has external calls inside a loop: token =
      IERC20MetadataUpgradeable(integrationMap.getTokenAddress(
      tokenId)) (contracts/yield-integrations/
      DynamicRangeOrdersIntegration.sol#372-374)
80 DynamicRangeOrdersIntegration.
      transferTokensTodynamicRangeOrdersIntegrationDeployer() (
      contracts/yield-integrations/DynamicRangeOrdersIntegration.sol
      #365-392) has external calls inside a loop: token.allowance(
      address(dynamicRangeOrdersIntegrationDeployer),address(this))
```

```
      == 0 (contracts/yield-integrations/
      DynamicRangeOrdersIntegration.sol#377-380)
81 DynamicRangeOrdersIntegration.
      transferTokensTodynamicRangeOrdersIntegrationDeployer() (
      contracts/yield-integrations/DynamicRangeOrdersIntegration.sol
      #365-392) has external calls inside a loop:
      dynamicRangeOrdersIntegrationDeployer.tokenApprovals(address(
      token)) (contracts/yield-integrations/
      DynamicRangeOrdersIntegration.sol#382)
82 DynamicRangeOrdersIntegration.
      transferTokensTodynamicRangeOrdersIntegrationDeployer() (
      contracts/yield-integrations/DynamicRangeOrdersIntegration.sol
      #365-392) has external calls inside a loop: token.balanceOf(
      address(this)) > 0 (contracts/yield-integrations/
      DynamicRangeOrdersIntegration.sol#385)
83 DynamicRangeOrdersIntegration.
      transferTokensTodynamicRangeOrdersIntegrationDeployer() (
      contracts/yield-integrations/DynamicRangeOrdersIntegration.sol
      #365-392) has external calls inside a loop: token.safeTransfer(
      address(dynamicRangeOrdersIntegrationDeployer),token.balanceOf(
      address(this))) (contracts/yield-integrations/
      DynamicRangeOrdersIntegration.sol#386-389)
84 DynamicRangeOrdersIntegration.increaseLiquidityPositions(uint256
      []) (contracts/yield-integrations/DynamicRangeOrdersIntegration
      .sol#395-465) has external calls inside a loop:
      dynamicRangeOrdersIntegrationDeployer.increaseLiquidityPosition
      (liquidityPositions[liquidityPositionKeys[
      liquidityPositionIndex]].id,liquidityPositions[
      liquidityPositionKeys[liquidityPositionIndex]].token0,
      liquidityPositions[liquidityPositionKeys[liquidityPositionIndex
      ]].token1,amount0Desired,amount1Desired) (contracts/yield-
      integrations/DynamicRangeOrdersIntegration.sol#415-424)
85 DynamicRangeOrdersIntegration.increaseLiquidityPositions(uint256
      []) (contracts/yield-integrations/DynamicRangeOrdersIntegration
      .sol#395-465) has external calls inside a loop: (success,
      liquidityPositionId) = dynamicRangeOrdersIntegrationDeployer.
      mintLiquidityPosition(liquidityPositions[liquidityPositionKeys[
      liquidityPositionIndex]].token0,liquidityPositions[
      liquidityPositionKeys[liquidityPositionIndex]].token1,
      liquidityPositions[liquidityPositionKeys[liquidityPositionIndex
      ]].feeNumerator,liquidityPositions[liquidityPositionKeys[
      liquidityPositionIndex]].tickLower,liquidityPositions[
      liquidityPositionKeys[liquidityPositionIndex]].tickUpper,
      amount0Desired_scope_0,amount1Desired_scope_1) (contracts/yield
```

```
      -integrations/DynamicRangeOrdersIntegration.sol#437-453)
86 DynamicRangeOrdersIntegration.closeExcessLiquidityPositions() (
      contracts/yield-integrations/DynamicRangeOrdersIntegration.sol
      #521-547) has external calls inside a loop:
      dynamicRangeOrdersIntegrationDeployer.decreaseLiquidityPosition
      (liquidityPositionKeys[liquidityPositionIndex],
      positionActualBaseStablecoinValue -
      positionDesiredBaseStablecoinValue) (contracts/yield-
      integrations/DynamicRangeOrdersIntegration.sol#541-544)
87 DynamicRangeOrdersIntegration.getBaseStablecoinReserveBalance() (
      contracts/yield-integrations/DynamicRangeOrdersIntegration.sol
      #600-629) has external calls inside a loop: tokenAddress =
      integrationMap.getTokenAddress(tokenId) (contracts/yield-
      integrations/DynamicRangeOrdersIntegration.sol#610)
88 DynamicRangeOrdersIntegration.getBaseStablecoinReserveBalance() (
      contracts/yield-integrations/DynamicRangeOrdersIntegration.sol
      #600-629) has external calls inside a loop: IUniswapTrader(
      moduleMap.getModuleAddress(Modules.UniswapTrader)).
      getTokenPairPoolsLength(tokenAddress,getBaseStablecoinAddress()
      ) > 0 (contracts/yield-integrations/
      DynamicRangeOrdersIntegration.sol#613-615)
89 DynamicRangeOrdersIntegration.getBaseStablecoinReserveBalance() (
      contracts/yield-integrations/DynamicRangeOrdersIntegration.sol
      #600-629) has external calls inside a loop:
      baseStablecoinReserveBalance += getTokenValueInBaseStablecoin(
      tokenAddress,IERC20MetadataUpgradeable(tokenAddress).balanceOf(
      address(dynamicRangeOrdersIntegrationDeployer))) (contracts/
      yield-integrations/DynamicRangeOrdersIntegration.sol#617-622)
90 DynamicRangeOrdersIntegration.getBaseStablecoinReserveBalance() (
      contracts/yield-integrations/DynamicRangeOrdersIntegration.sol
      #600-629) has external calls inside a loop:
      baseStablecoinReserveBalance += IERC20MetadataUpgradeable(
      tokenAddress).balanceOf(address(
      dynamicRangeOrdersIntegrationDeployer)) (contracts/yield-
      integrations/DynamicRangeOrdersIntegration.sol#625-626)
91 DynamicRangeOrdersIntegrationDeployer.
      swapExcessTokensForBaseStablecoin(uint256[]) (contracts/yield-
      integrations/DynamicRangeOrdersIntegrationDeployer.sol#85-120)
      has external calls inside a loop: tokenAddress = integrationMap
      .getTokenAddress(tokenId) (contracts/yield-integrations/
      DynamicRangeOrdersIntegrationDeployer.sol#98)
92 DynamicRangeOrdersIntegrationDeployer.
      swapExcessTokensForBaseStablecoin(uint256[]) (contracts/yield-
      integrations/DynamicRangeOrdersIntegrationDeployer.sol#85-120)
```

```
      has external calls inside a loop: tokenBalance = token.
      balanceOf(address(this)) (contracts/yield-integrations/
      DynamicRangeOrdersIntegrationDeployer.sol#104)
93 DynamicRangeOrdersIntegrationDeployer.
      swapExcessTokensForBaseStablecoin(uint256[]) (contracts/yield-
      integrations/DynamicRangeOrdersIntegrationDeployer.sol#85-120)
      has external calls inside a loop: uniswapTrader.swapExactInput(
      tokenAddress,baseStablecoinAddress,address(this),tokenBalance -
       tokenDesiredAmounts[tokenId]) (contracts/yield-integrations/
      DynamicRangeOrdersIntegrationDeployer.sol#111-116)
94 DynamicRangeOrdersIntegrationDeployer.
      swapExcessBaseStablecoinForTokens(uint256[]) (contracts/yield-
      integrations/DynamicRangeOrdersIntegrationDeployer.sol#123-173)
       has external calls inside a loop: integrationMap.
      getTokenAddress(tokenId) != baseStablecoinAddress (contracts/
      yield-integrations/DynamicRangeOrdersIntegrationDeployer.sol
      #136)
95 DynamicRangeOrdersIntegrationDeployer.
      swapExcessBaseStablecoinForTokens(uint256[]) (contracts/yield-
      integrations/DynamicRangeOrdersIntegrationDeployer.sol#123-173)
       has external calls inside a loop: tokenAddress =
      integrationMap.getTokenAddress(tokenId) (contracts/yield-
      integrations/DynamicRangeOrdersIntegrationDeployer.sol#137)
96 DynamicRangeOrdersIntegrationDeployer.
      swapExcessBaseStablecoinForTokens(uint256[]) (contracts/yield-
      integrations/DynamicRangeOrdersIntegrationDeployer.sol#123-173)
       has external calls inside a loop: tokenBalance = token.
      balanceOf(address(this)) (contracts/yield-integrations/
      DynamicRangeOrdersIntegrationDeployer.sol#141)
97 DynamicRangeOrdersIntegrationDeployer.
      swapExcessBaseStablecoinForTokens(uint256[]) (contracts/yield-
      integrations/DynamicRangeOrdersIntegrationDeployer.sol#123-173)
       has external calls inside a loop: baseStablecoinAmount =
      dynamicRangeOrdersIntegration.getTokenValueInBaseStablecoin(
      tokenAddress,tokenDesiredAmounts[tokenId] - tokenBalance) (
      contracts/yield-integrations/
      DynamicRangeOrdersIntegrationDeployer.sol#144-148)
98 DynamicRangeOrdersIntegrationDeployer.
      swapExcessBaseStablecoinForTokens(uint256[]) (contracts/yield-
      integrations/DynamicRangeOrdersIntegrationDeployer.sol#123-173)
       has external calls inside a loop: baseStablecoinAmount >
      IERC20MetadataUpgradeable(baseStablecoinAddress).balanceOf(
      address(this)) (contracts/yield-integrations/
      DynamicRangeOrdersIntegrationDeployer.sol#150-153)
```

```
 99 DynamicRangeOrdersIntegrationDeployer.
      swapExcessBaseStablecoinForTokens(uint256[]) (contracts/yield-
      integrations/DynamicRangeOrdersIntegrationDeployer.sol#123-173)
       has external calls inside a loop: baseStablecoinAmount =
      IERC20MetadataUpgradeable(baseStablecoinAddress).balanceOf(
      address(this)) (contracts/yield-integrations/
      DynamicRangeOrdersIntegrationDeployer.sol#155-157)
100 DynamicRangeOrdersIntegrationDeployer.
      swapExcessBaseStablecoinForTokens(uint256[]) (contracts/yield-
      integrations/DynamicRangeOrdersIntegrationDeployer.sol#123-173)
       has external calls inside a loop: uniswapTrader.swapExactInput
      (baseStablecoinAddress,tokenAddress,address(this),
      baseStablecoinAmount) (contracts/yield-integrations/
      DynamicRangeOrdersIntegrationDeployer.sol#164-169)
101 DynamicRangeOrdersIntegrationDeployer.closePositionsForWithdrawal(
      bytes32[],address,uint256) (contracts/yield-integrations/
      DynamicRangeOrdersIntegrationDeployer.sol#296-373) has external
       calls inside a loop: dynamicRangeOrdersIntegration.
      getPositionBaseStablecoinValue(liquidityPositionKeys[
      liquidityPositionIndex]) + dynamicRangeOrdersIntegration.
      getTokenValueInBaseStablecoin(tokenAddress,token.balanceOf(
      address(this))) <= withdrawalAmountInBaseStablecoinValue (
      contracts/yield-integrations/
      DynamicRangeOrdersIntegrationDeployer.sol#310-317)
102 DynamicRangeOrdersIntegrationDeployer.closePositionsForWithdrawal(
      bytes32[],address,uint256) (contracts/yield-integrations/
      DynamicRangeOrdersIntegrationDeployer.sol#296-373) has external
       calls inside a loop: decreaseLiquidityPosition(
      liquidityPositionKeys[liquidityPositionIndex],
      dynamicRangeOrdersIntegration.getPositionBaseStablecoinValue(
      liquidityPositionKeys[liquidityPositionIndex])) (contracts/
      yield-integrations/DynamicRangeOrdersIntegrationDeployer.sol
      #320-325)
103 DynamicRangeOrdersIntegrationDeployer.closePositionsForWithdrawal(
      bytes32[],address,uint256) (contracts/yield-integrations/
      DynamicRangeOrdersIntegrationDeployer.sol#296-373) has external
       calls inside a loop: token.balanceOf(address(this)) >= amount
      (contracts/yield-integrations/
      DynamicRangeOrdersIntegrationDeployer.sol#327)
104 DynamicRangeOrdersIntegrationDeployer.closePositionsForWithdrawal(
      bytes32[],address,uint256) (contracts/yield-integrations/
      DynamicRangeOrdersIntegrationDeployer.sol#296-373) has external
       calls inside a loop: liquidityPositionIndex ==
      dynamicRangeOrdersIntegration.getLiquidityPositionsCount() - 1
```

```
      (contracts/yield-integrations/
      DynamicRangeOrdersIntegrationDeployer.sol#364-365)
105 DynamicRangeOrdersIntegrationDeployer.closePositionsForWithdrawal(
      bytes32[],address,uint256) (contracts/yield-integrations/
      DynamicRangeOrdersIntegrationDeployer.sol#296-373) has external
       calls inside a loop: decreaseLiquidityPosition(
      liquidityPositionKeys[liquidityPositionIndex],(102 * (
      withdrawalAmountInBaseStablecoinValue -
      dynamicRangeOrdersIntegration.getTokenValueInBaseStablecoin(
      tokenAddress,token.balanceOf(address(this))))) / 100) (
      contracts/yield-integrations/
      DynamicRangeOrdersIntegrationDeployer.sol#332-340)
106 DynamicRangeOrdersIntegrationDeployer.closePositionsForWithdrawal(
      bytes32[],address,uint256) (contracts/yield-integrations/
      DynamicRangeOrdersIntegrationDeployer.sol#296-373) has external
       calls inside a loop: token.balanceOf(address(this)) >= amount
      (contracts/yield-integrations/
      DynamicRangeOrdersIntegrationDeployer.sol#344)
107 DynamicRangeOrdersIntegrationDeployer.closePositionsForWithdrawal(
      bytes32[],address,uint256) (contracts/yield-integrations/
      DynamicRangeOrdersIntegrationDeployer.sol#296-373) has external
       calls inside a loop: decreaseLiquidityPosition(
      liquidityPositionKeys[liquidityPositionIndex],
      dynamicRangeOrdersIntegration.getPositionBaseStablecoinValue(
      liquidityPositionKeys[liquidityPositionIndex])) (contracts/
      yield-integrations/DynamicRangeOrdersIntegrationDeployer.sol
      #349-354)
108 DynamicRangeOrdersIntegrationDeployer.closePositionsForWithdrawal(
      bytes32[],address,uint256) (contracts/yield-integrations/
      DynamicRangeOrdersIntegrationDeployer.sol#296-373) has external
       calls inside a loop: token.balanceOf(address(this)) >= amount
      (contracts/yield-integrations/
      DynamicRangeOrdersIntegrationDeployer.sol#356)
109 DynamicRangeOrdersIntegrationDeployer.harvestYield(bytes32[]) (
      contracts/yield-integrations/
      DynamicRangeOrdersIntegrationDeployer.sol#485-618) has external
       calls inside a loop: liquidityPositionIndex <
      dynamicRangeOrdersIntegration.getLiquidityPositionsCount() (
      contracts/yield-integrations/
      DynamicRangeOrdersIntegrationDeployer.sol#506-507)
110 DynamicRangeOrdersIntegrationDeployer.harvestYield(bytes32[]) (
      contracts/yield-integrations/
      DynamicRangeOrdersIntegrationDeployer.sol#485-618) has external
       calls inside a loop: (token0,token1,liquidityPositionId) =
```

```
      dynamicRangeOrdersIntegration.getLiquidityPosition(
      liquidityPositionKeys[liquidityPositionIndex]) (contracts/yield
      -integrations/DynamicRangeOrdersIntegrationDeployer.sol
      #510-521)
111 DynamicRangeOrdersIntegrationDeployer.harvestYield(bytes32[]) (
      contracts/yield-integrations/
      DynamicRangeOrdersIntegrationDeployer.sol#485-618) has external
       calls inside a loop: token0Id = integrationMap.getTokenId(
      token0) (contracts/yield-integrations/
      DynamicRangeOrdersIntegrationDeployer.sol#523)
112 DynamicRangeOrdersIntegrationDeployer.harvestYield(bytes32[]) (
      contracts/yield-integrations/
      DynamicRangeOrdersIntegrationDeployer.sol#485-618) has external
       calls inside a loop: token1Id = integrationMap.getTokenId(
      token1) (contracts/yield-integrations/
      DynamicRangeOrdersIntegrationDeployer.sol#524)
113 DynamicRangeOrdersIntegrationDeployer.harvestYield(bytes32[]) (
      contracts/yield-integrations/
      DynamicRangeOrdersIntegrationDeployer.sol#485-618) has external
       calls inside a loop: tokenAddress = integrationMap.
      getTokenAddress(tokenId) (contracts/yield-integrations/
      DynamicRangeOrdersIntegrationDeployer.sol#546)
114 DynamicRangeOrdersIntegrationDeployer.harvestYield(bytes32[]) (
      contracts/yield-integrations/
      DynamicRangeOrdersIntegrationDeployer.sol#485-618) has external
       calls inside a loop: tokenAddress == baseStablecoinAddress ||
      IUniswapTrader(moduleMap.getModuleAddress(Modules.UniswapTrader
      )).getTokenPairPoolsLength(tokenAddress,baseStablecoinAddress)
      > 0 (contracts/yield-integrations/
      DynamicRangeOrdersIntegrationDeployer.sol#550-553)
115 DynamicRangeOrdersIntegrationDeployer.harvestYield(bytes32[]) (
      contracts/yield-integrations/
      DynamicRangeOrdersIntegrationDeployer.sol#485-618) has external
       calls inside a loop: tokenBalanceBaseStablecoinValue =
      dynamicRangeOrdersIntegration.getTokenValueInBaseStablecoin(
      tokenAddress,dynamicRangeOrdersIntegration.getBalance(
      tokenAddress)) (contracts/yield-integrations/
      DynamicRangeOrdersIntegrationDeployer.sol#555-559)
116 DynamicRangeOrdersIntegrationDeployer.harvestYield(bytes32[]) (
      contracts/yield-integrations/
      DynamicRangeOrdersIntegrationDeployer.sol#485-618) has external
       calls inside a loop: IERC20MetadataUpgradeable(tokenAddress).
      safeTransfer(moduleMap.getModuleAddress(Modules.UniswapTrader),
      tokensHarvestedYield[tokenId]) (contracts/yield-integrations/
```

```
      DynamicRangeOrdersIntegrationDeployer.sol#566-569)
117 DynamicRangeOrdersIntegrationDeployer.harvestYield(bytes32[]) (
      contracts/yield-integrations/
      DynamicRangeOrdersIntegrationDeployer.sol#485-618) has external
       calls inside a loop: IUniswapTrader(moduleMap.getModuleAddress
      (Modules.UniswapTrader)).swapExactInput(tokenAddress,
      baseStablecoinAddress,address(this),tokensHarvestedYield[
      tokenId]) (contracts/yield-integrations/
      DynamicRangeOrdersIntegrationDeployer.sol#570-576)
118 DynamicRangeOrdersIntegrationDeployer.harvestYield(bytes32[]) (
      contracts/yield-integrations/
      DynamicRangeOrdersIntegrationDeployer.sol#485-618) has external
       calls inside a loop: tokenAddress_scope_1 = integrationMap.
      getTokenAddress(tokenId_scope_0) (contracts/yield-integrations/
      DynamicRangeOrdersIntegrationDeployer.sol#590)
119 DynamicRangeOrdersIntegrationDeployer.harvestYield(bytes32[]) (
      contracts/yield-integrations/
      DynamicRangeOrdersIntegrationDeployer.sol#485-618) has external
       calls inside a loop: IERC20MetadataUpgradeable(
      baseStablecoinAddress).safeTransfer(moduleMap.getModuleAddress(
      Modules.UniswapTrader),baseStablecoinAmount) (contracts/yield-
      integrations/DynamicRangeOrdersIntegrationDeployer.sol#597-600)
120 DynamicRangeOrdersIntegrationDeployer.harvestYield(bytes32[]) (
      contracts/yield-integrations/
      DynamicRangeOrdersIntegrationDeployer.sol#485-618) has external
       calls inside a loop: IUniswapTrader(moduleMap.getModuleAddress
      (Modules.UniswapTrader)).swapExactInput(baseStablecoinAddress,
      tokenAddress_scope_1,moduleMap.getModuleAddress(Modules.
      YieldManager),baseStablecoinAmount) (contracts/yield-
      integrations/DynamicRangeOrdersIntegrationDeployer.sol#602-608)
121 DynamicRangeOrdersIntegrationDeployer.harvestYield(bytes32[]) (
      contracts/yield-integrations/
      DynamicRangeOrdersIntegrationDeployer.sol#485-618) has external
       calls inside a loop: IERC20MetadataUpgradeable(
      baseStablecoinAddress).safeTransfer(moduleMap.getModuleAddress(
      Modules.YieldManager),baseStablecoinAmount) (contracts/yield-
      integrations/DynamicRangeOrdersIntegrationDeployer.sol#610-613)
122 SushiSwapIntegration.withdraw(address,uint256) (contracts/yield-
      integrations/SushiSwapIntegration.sol#307-333) has external
      calls inside a loop: IERC20MetadataUpgradeable(token).
      safeTransfer(moduleMap.getModuleAddress(Modules.Kernel),
      ethReserve - (eth - amount)) (contracts/yield-integrations/
      SushiSwapIntegration.sol#319)
```

```
123 SushiSwapIntegration.withdraw(address,uint256) (contracts/yield-
      integrations/SushiSwapIntegration.sol#307-333) has external
      calls inside a loop: IERC20MetadataUpgradeable(token).
      safeTransfer(moduleMap.getModuleAddress(Modules.Kernel),
      ethReserve) (contracts/yield-integrations/SushiSwapIntegration.
      sol#319)
124 UniswapIntegration._deploy() (contracts/yield-integrations/
      UniswapIntegration.sol#282-349) has external calls inside a
      loop: token0Id = integrationMap.getTokenId(liquidityPositions[
      liquidityPositionIndex].token0) (contracts/yield-integrations/
      UniswapIntegration.sol#320-322)
125 UniswapIntegration._deploy() (contracts/yield-integrations/
      UniswapIntegration.sol#282-349) has external calls inside a
      loop: token1Id = integrationMap.getTokenId(liquidityPositions[
      liquidityPositionIndex].token1) (contracts/yield-integrations/
      UniswapIntegration.sol#323-325)
126 UniswapIntegration.transferTokensToUniswapIntegrationDeployer() (
      contracts/yield-integrations/UniswapIntegration.sol#377-402)
      has external calls inside a loop: token =
      IERC20MetadataUpgradeable(integrationMap.getTokenAddress(
      tokenId)) (contracts/yield-integrations/UniswapIntegration.sol
      #384-386)
127 UniswapIntegration.transferTokensToUniswapIntegrationDeployer() (
      contracts/yield-integrations/UniswapIntegration.sol#377-402)
      has external calls inside a loop: token.balanceOf(address(this)
      ) > 0 (contracts/yield-integrations/UniswapIntegration.sol#388)
128 UniswapIntegration.transferTokensToUniswapIntegrationDeployer() (
      contracts/yield-integrations/UniswapIntegration.sol#377-402)
      has external calls inside a loop: token.allowance(address(
      uniswapIntegrationDeployer),address(this)) == 0 (contracts/
      yield-integrations/UniswapIntegration.sol#390-391)
129 UniswapIntegration.transferTokensToUniswapIntegrationDeployer() (
      contracts/yield-integrations/UniswapIntegration.sol#377-402)
      has external calls inside a loop: uniswapIntegrationDeployer.
      tokenApprovals(address(token)) (contracts/yield-integrations/
      UniswapIntegration.sol#393)
130 UniswapIntegration.transferTokensToUniswapIntegrationDeployer() (
      contracts/yield-integrations/UniswapIntegration.sol#377-402)
      has external calls inside a loop: token.safeTransfer(address(
      uniswapIntegrationDeployer),token.balanceOf(address(this))) (
      contracts/yield-integrations/UniswapIntegration.sol#396-399)
131 UniswapIntegration.increaseLiquidityPositions(uint256[]) (
      contracts/yield-integrations/UniswapIntegration.sol#405-464)
      has external calls inside a loop: uniswapIntegrationDeployer.
```

```
      increaseLiquidityPosition(liquidityPositions[
      liquidityPositionIndex].id,liquidityPositions[
      liquidityPositionIndex].token0,liquidityPositions[
      liquidityPositionIndex].token1,amount0Desired,amount1Desired) (
      contracts/yield-integrations/UniswapIntegration.sol#424-430)
132 UniswapIntegration.increaseLiquidityPositions(uint256[]) (
      contracts/yield-integrations/UniswapIntegration.sol#405-464)
      has external calls inside a loop: (success,liquidityPositionId)
       = uniswapIntegrationDeployer.mintLiquidityPosition(
      liquidityPositions[liquidityPositionIndex].token0,
      liquidityPositions[liquidityPositionIndex].token1,
      liquidityPositions[liquidityPositionIndex].feeNumerator,
      liquidityPositions[liquidityPositionIndex].tickLower,
      liquidityPositions[liquidityPositionIndex].tickUpper,
      amount0Desired_scope_0,amount1Desired_scope_1) (contracts/yield
      -integrations/UniswapIntegration.sol#443-454)
133 UniswapIntegration.closeExcessLiquidityPositions() (contracts/
      yield-integrations/UniswapIntegration.sol#467-493) has external
       calls inside a loop: uniswapIntegrationDeployer.
      decreaseLiquidityPosition(liquidityPositionIndex,
      positionActualBaseStablecoinValue -
      positionDesiredBaseStablecoinValue) (contracts/yield-
      integrations/UniswapIntegration.sol#487-490)
134 UniswapIntegration.getBaseStablecoinReserveBalance() (contracts/
      yield-integrations/UniswapIntegration.sol#546-575) has external
       calls inside a loop: tokenAddress = integrationMap.
      getTokenAddress(tokenId) (contracts/yield-integrations/
      UniswapIntegration.sol#556)
135 UniswapIntegration.getBaseStablecoinReserveBalance() (contracts/
      yield-integrations/UniswapIntegration.sol#546-575) has external
       calls inside a loop: IUniswapTrader(moduleMap.getModuleAddress
      (Modules.UniswapTrader)).getTokenPairPoolsLength(tokenAddress,
      getBaseStablecoinAddress()) > 0 (contracts/yield-integrations/
      UniswapIntegration.sol#559-561)
136 UniswapIntegration.getBaseStablecoinReserveBalance() (contracts/
      yield-integrations/UniswapIntegration.sol#546-575) has external
       calls inside a loop: baseStablecoinReserveBalance +=
      getTokenValueInBaseStablecoin(tokenAddress,
      IERC20MetadataUpgradeable(tokenAddress).balanceOf(address(
      uniswapIntegrationDeployer))) (contracts/yield-integrations/
      UniswapIntegration.sol#563-568)
137 UniswapIntegration.getBaseStablecoinReserveBalance() (contracts/
      yield-integrations/UniswapIntegration.sol#546-575) has external
       calls inside a loop: baseStablecoinReserveBalance +=
```

```
      IERC20MetadataUpgradeable(tokenAddress).balanceOf(address(
      uniswapIntegrationDeployer)) (contracts/yield-integrations/
      UniswapIntegration.sol#571-572)
138 UniswapIntegrationDeployer.swapExcessTokensForBaseStablecoin(
      uint256[]) (contracts/yield-integrations/
      UniswapIntegrationDeployer.sol#83-118) has external calls
      inside a loop: tokenAddress = integrationMap.getTokenAddress(
      tokenId) (contracts/yield-integrations/
      UniswapIntegrationDeployer.sol#96)
139 UniswapIntegrationDeployer.swapExcessTokensForBaseStablecoin(
      uint256[]) (contracts/yield-integrations/
      UniswapIntegrationDeployer.sol#83-118) has external calls
      inside a loop: tokenBalance = token.balanceOf(address(this)) (
      contracts/yield-integrations/UniswapIntegrationDeployer.sol
      #102)
140 UniswapIntegrationDeployer.swapExcessTokensForBaseStablecoin(
      uint256[]) (contracts/yield-integrations/
      UniswapIntegrationDeployer.sol#83-118) has external calls
      inside a loop: uniswapTrader.swapExactInput(tokenAddress,
      baseStablecoinAddress,address(this),tokenBalance -
      tokenDesiredAmounts[tokenId]) (contracts/yield-integrations/
      UniswapIntegrationDeployer.sol#109-114)
141 UniswapIntegrationDeployer.swapExcessBaseStablecoinForTokens(
      uint256[]) (contracts/yield-integrations/
      UniswapIntegrationDeployer.sol#121-171) has external calls
      inside a loop: integrationMap.getTokenAddress(tokenId) !=
      baseStablecoinAddress (contracts/yield-integrations/
      UniswapIntegrationDeployer.sol#134)
142 UniswapIntegrationDeployer.swapExcessBaseStablecoinForTokens(
      uint256[]) (contracts/yield-integrations/
      UniswapIntegrationDeployer.sol#121-171) has external calls
      inside a loop: tokenAddress = integrationMap.getTokenAddress(
      tokenId) (contracts/yield-integrations/
      UniswapIntegrationDeployer.sol#135)
143 UniswapIntegrationDeployer.swapExcessBaseStablecoinForTokens(
      uint256[]) (contracts/yield-integrations/
      UniswapIntegrationDeployer.sol#121-171) has external calls
      inside a loop: tokenBalance = token.balanceOf(address(this)) (
      contracts/yield-integrations/UniswapIntegrationDeployer.sol
      #139)
144 UniswapIntegrationDeployer.swapExcessBaseStablecoinForTokens(
      uint256[]) (contracts/yield-integrations/
      UniswapIntegrationDeployer.sol#121-171) has external calls
      inside a loop: baseStablecoinAmount = uniswapIntegration.
```

```
      getTokenValueInBaseStablecoin(tokenAddress,tokenDesiredAmounts[
      tokenId] - tokenBalance) (contracts/yield-integrations/
      UniswapIntegrationDeployer.sol#142-146)
145 UniswapIntegrationDeployer.swapExcessBaseStablecoinForTokens(
      uint256[]) (contracts/yield-integrations/
      UniswapIntegrationDeployer.sol#121-171) has external calls
      inside a loop: baseStablecoinAmount > IERC20MetadataUpgradeable
      (baseStablecoinAddress).balanceOf(address(this)) (contracts/
      yield-integrations/UniswapIntegrationDeployer.sol#148-151)
146 UniswapIntegrationDeployer.swapExcessBaseStablecoinForTokens(
      uint256[]) (contracts/yield-integrations/
      UniswapIntegrationDeployer.sol#121-171) has external calls
      inside a loop: baseStablecoinAmount = IERC20MetadataUpgradeable
      (baseStablecoinAddress).balanceOf(address(this)) (contracts/
      yield-integrations/UniswapIntegrationDeployer.sol#153-155)
147 UniswapIntegrationDeployer.swapExcessBaseStablecoinForTokens(
      uint256[]) (contracts/yield-integrations/
      UniswapIntegrationDeployer.sol#121-171) has external calls
      inside a loop: uniswapTrader.swapExactInput(
      baseStablecoinAddress,tokenAddress,address(this),
      baseStablecoinAmount) (contracts/yield-integrations/
      UniswapIntegrationDeployer.sol#162-167)
148 UniswapIntegrationDeployer.closePositionsForWithdrawal(address,
      uint256) (contracts/yield-integrations/
      UniswapIntegrationDeployer.sol#294-371) has external calls
      inside a loop: uniswapIntegration.
      getPositionBaseStablecoinValue(liquidityPositionIndex) +
      uniswapIntegration.getTokenValueInBaseStablecoin(tokenAddress,
      token.balanceOf(address(this))) <=
      withdrawalAmountInBaseStablecoinValue (contracts/yield-
      integrations/UniswapIntegrationDeployer.sol#308-315)
149 UniswapIntegrationDeployer.closePositionsForWithdrawal(address,
      uint256) (contracts/yield-integrations/
      UniswapIntegrationDeployer.sol#294-371) has external calls
      inside a loop: decreaseLiquidityPosition(liquidityPositionIndex
      ,uniswapIntegration.getPositionBaseStablecoinValue(
      liquidityPositionIndex)) (contracts/yield-integrations/
      UniswapIntegrationDeployer.sol#318-323)
150 UniswapIntegrationDeployer.closePositionsForWithdrawal(address,
      uint256) (contracts/yield-integrations/
      UniswapIntegrationDeployer.sol#294-371) has external calls
      inside a loop: token.balanceOf(address(this)) >= amount (
      contracts/yield-integrations/UniswapIntegrationDeployer.sol
      #325)
```

```
151 UniswapIntegrationDeployer.closePositionsForWithdrawal(address,
      uint256) (contracts/yield-integrations/
      UniswapIntegrationDeployer.sol#294-371) has external calls
      inside a loop: liquidityPositionIndex == uniswapIntegration.
      getLiquidityPositionsCount() - 1 (contracts/yield-integrations/
      UniswapIntegrationDeployer.sol#362-363)
152 UniswapIntegrationDeployer.closePositionsForWithdrawal(address,
      uint256) (contracts/yield-integrations/
      UniswapIntegrationDeployer.sol#294-371) has external calls
      inside a loop: decreaseLiquidityPosition(liquidityPositionIndex
      ,(102 * (withdrawalAmountInBaseStablecoinValue -
      uniswapIntegration.getTokenValueInBaseStablecoin(tokenAddress,
      token.balanceOf(address(this))))) / 100) (contracts/yield-
      integrations/UniswapIntegrationDeployer.sol#330-338)
153 UniswapIntegrationDeployer.closePositionsForWithdrawal(address,
      uint256) (contracts/yield-integrations/
      UniswapIntegrationDeployer.sol#294-371) has external calls
      inside a loop: token.balanceOf(address(this)) >= amount (
      contracts/yield-integrations/UniswapIntegrationDeployer.sol
      #342)
154 UniswapIntegrationDeployer.closePositionsForWithdrawal(address,
      uint256) (contracts/yield-integrations/
      UniswapIntegrationDeployer.sol#294-371) has external calls
      inside a loop: decreaseLiquidityPosition(liquidityPositionIndex
      ,uniswapIntegration.getPositionBaseStablecoinValue(
      liquidityPositionIndex)) (contracts/yield-integrations/
      UniswapIntegrationDeployer.sol#347-352)
155 UniswapIntegrationDeployer.closePositionsForWithdrawal(address,
      uint256) (contracts/yield-integrations/
      UniswapIntegrationDeployer.sol#294-371) has external calls
      inside a loop: token.balanceOf(address(this)) >= amount (
      contracts/yield-integrations/UniswapIntegrationDeployer.sol
      #354)
156 UniswapIntegrationDeployer.harvestYield() (contracts/yield-
      integrations/UniswapIntegrationDeployer.sol#491-618) has
      external calls inside a loop: liquidityPositionIndex <
      uniswapIntegration.getLiquidityPositionsCount() (contracts/
      yield-integrations/UniswapIntegrationDeployer.sol#508)
157 UniswapIntegrationDeployer.harvestYield() (contracts/yield-
      integrations/UniswapIntegrationDeployer.sol#491-618) has
      external calls inside a loop: (token0,token1,minted,
      liquidityPositionId) = uniswapIntegration.getLiquidityPosition(
      liquidityPositionIndex) (contracts/yield-integrations/
      UniswapIntegrationDeployer.sol#511-520)
```

```
158 UniswapIntegrationDeployer.harvestYield() (contracts/yield-
      integrations/UniswapIntegrationDeployer.sol#491-618) has
      external calls inside a loop: token0Id = integrationMap.
      getTokenId(token0) (contracts/yield-integrations/
      UniswapIntegrationDeployer.sol#522)
159 UniswapIntegrationDeployer.harvestYield() (contracts/yield-
      integrations/UniswapIntegrationDeployer.sol#491-618) has
      external calls inside a loop: token1Id = integrationMap.
      getTokenId(token1) (contracts/yield-integrations/
      UniswapIntegrationDeployer.sol#523)
160 UniswapIntegrationDeployer.harvestYield() (contracts/yield-
      integrations/UniswapIntegrationDeployer.sol#491-618) has
      external calls inside a loop: tokenAddress = integrationMap.
      getTokenAddress(tokenId) (contracts/yield-integrations/
      UniswapIntegrationDeployer.sol#546)
161 UniswapIntegrationDeployer.harvestYield() (contracts/yield-
      integrations/UniswapIntegrationDeployer.sol#491-618) has
      external calls inside a loop: tokenAddress ==
      baseStablecoinAddress || IUniswapTrader(moduleMap.
      getModuleAddress(Modules.UniswapTrader)).
      getTokenPairPoolsLength(tokenAddress,baseStablecoinAddress) > 0
       (contracts/yield-integrations/UniswapIntegrationDeployer.sol
      #550-553)
162 UniswapIntegrationDeployer.harvestYield() (contracts/yield-
      integrations/UniswapIntegrationDeployer.sol#491-618) has
      external calls inside a loop: tokenBalanceBaseStablecoinValue =
       uniswapIntegration.getTokenValueInBaseStablecoin(tokenAddress,
      uniswapIntegration.getBalance(tokenAddress)) (contracts/yield-
      integrations/UniswapIntegrationDeployer.sol#555-559)
163 UniswapIntegrationDeployer.harvestYield() (contracts/yield-
      integrations/UniswapIntegrationDeployer.sol#491-618) has
      external calls inside a loop: IERC20MetadataUpgradeable(
      tokenAddress).safeTransfer(moduleMap.getModuleAddress(Modules.
      UniswapTrader),tokensHarvestedYield[tokenId]) (contracts/yield-
      integrations/UniswapIntegrationDeployer.sol#566-569)
164 UniswapIntegrationDeployer.harvestYield() (contracts/yield-
      integrations/UniswapIntegrationDeployer.sol#491-618) has
      external calls inside a loop: IUniswapTrader(moduleMap.
      getModuleAddress(Modules.UniswapTrader)).swapExactInput(
      tokenAddress,baseStablecoinAddress,address(this),
      tokensHarvestedYield[tokenId]) (contracts/yield-integrations/
      UniswapIntegrationDeployer.sol#570-576)
165 UniswapIntegrationDeployer.harvestYield() (contracts/yield-
      integrations/UniswapIntegrationDeployer.sol#491-618) has
```

```
      external calls inside a loop: tokenAddress_scope_1 =
      integrationMap.getTokenAddress(tokenId_scope_0) (contracts/
      yield-integrations/UniswapIntegrationDeployer.sol#590)
166 UniswapIntegrationDeployer.harvestYield() (contracts/yield-
      integrations/UniswapIntegrationDeployer.sol#491-618) has
      external calls inside a loop: IERC20MetadataUpgradeable(
      baseStablecoinAddress).safeTransfer(moduleMap.getModuleAddress(
      Modules.UniswapTrader),baseStablecoinAmount) (contracts/yield-
      integrations/UniswapIntegrationDeployer.sol#597-600)
167 UniswapIntegrationDeployer.harvestYield() (contracts/yield-
      integrations/UniswapIntegrationDeployer.sol#491-618) has
      external calls inside a loop: IUniswapTrader(moduleMap.
      getModuleAddress(Modules.UniswapTrader)).swapExactInput(
      baseStablecoinAddress,tokenAddress_scope_1,moduleMap.
      getModuleAddress(Modules.YieldManager),baseStablecoinAmount) (
      contracts/yield-integrations/UniswapIntegrationDeployer.sol
      #602-608)
168 UniswapIntegrationDeployer.harvestYield() (contracts/yield-
      integrations/UniswapIntegrationDeployer.sol#491-618) has
      external calls inside a loop: IERC20MetadataUpgradeable(
      baseStablecoinAddress).safeTransfer(moduleMap.getModuleAddress(
      Modules.YieldManager),baseStablecoinAmount) (contracts/yield-
      integrations/UniswapIntegrationDeployer.sol#610-613)
169 YearnIntegration.deploy() (contracts/yield-integrations/
      YearnIntegration.sol#101-124) has external calls inside a loop:
       token = IERC20MetadataUpgradeable(integrationMap.
      getTokenAddress(tokenId)) (contracts/yield-integrations/
      YearnIntegration.sol#108-110)
170 YearnIntegration.deploy() (contracts/yield-integrations/
      YearnIntegration.sol#101-124) has external calls inside a loop:
       tokenAmount = token.balanceOf(address(this)) (contracts/yield-
      integrations/YearnIntegration.sol#111)
171 YearnIntegration.deploy() (contracts/yield-integrations/
      YearnIntegration.sol#101-124) has external calls inside a loop:
       token.allowance(address(this),vaultAddress) == 0 (contracts/
      yield-integrations/YearnIntegration.sol#116)
172 YearnIntegration.deploy() (contracts/yield-integrations/
      YearnIntegration.sol#101-124) has external calls inside a loop:
       IYearnVault(vaultAddress).deposit(tokenAmount,address(this)) (
      contracts/yield-integrations/YearnIntegration.sol#121-123)
173 YearnIntegration.harvestYield() (contracts/yield-integrations/
      YearnIntegration.sol#130-166) has external calls inside a loop:
       token = IERC20MetadataUpgradeable(integrationMap.
      getTokenAddress(tokenId)) (contracts/yield-integrations/
```

```
      YearnIntegration.sol#137-139)
174 YearnIntegration.harvestYield() (contracts/yield-integrations/
      YearnIntegration.sol#130-166) has external calls inside a loop:
       balanceBefore = token.balanceOf(address(this)) (contracts/
      yield-integrations/YearnIntegration.sol#148-149)
175 YearnIntegration.harvestYield() (contracts/yield-integrations/
      YearnIntegration.sol#130-166) has external calls inside a loop:
       IYearnVault(getVaultAddress(address(token))).withdraw(
      availableYieldInShares) (contracts/yield-integrations/
      YearnIntegration.sol#151-164)
176 YearnIntegration.harvestYield() (contracts/yield-integrations/
      YearnIntegration.sol#130-166) has external calls inside a loop:
       harvestedAmount = token.balanceOf(address(this)) -
      balanceBefore (contracts/yield-integrations/YearnIntegration.
      sol#154-157)
177 YearnIntegration.harvestYield() (contracts/yield-integrations/
      YearnIntegration.sol#130-166) has external calls inside a loop:
       token.safeTransfer(moduleMap.getModuleAddress(Modules.
      YieldManager),harvestedAmount) (contracts/yield-integrations/
      YearnIntegration.sol#160-163)
```

Risk Level:

Likelihood - 2
Impact - 3

Recommendation:

If possible, use pull over push strategy for external calls.

Remediation Plan:

ACKNOWLEDGED: 0x_nodes acknowledged the issue and they will fix it later.

3.7 (HAL-07) IGNORED RETURN VALUES - LOW

Description:

The return value of an external call is not stored in a local or state variable. In the project there are many instances where external methods are being called and return values are being ignored.

Code Location:

Listing 25: Kernel.sol (Lines 475)

```
459   function withdraw(
460     address[] memory tokens,
461     uint256[] memory amounts,
462     bool withdrawWethAsEth
463   ) external {
464     uint256 ethWithdrawn = IUserPositions(
465       moduleMap.getModuleAddress(Modules.UserPositions)
466     ).withdraw(msg.sender, tokens, amounts, withdrawWethAsEth);
467
468     if (ethWithdrawn > 0) {
469       IWeth9(
470         IIntegrationMap(moduleMap.getModuleAddress(Modules.
              IntegrationMap))
471           .getWethTokenAddress()
472       ).withdraw(ethWithdrawn);
473
474
475       payable(msg.sender).transfer(ethWithdrawn);
476     }
477
478     emit Withdraw(msg.sender, tokens, amounts, ethWithdrawn);
479   }
480
```

Listing 26: Kernel.sol (Lines 510)

```
488   function withdrawAllAndClaim(address[] memory tokens, bool
          withdrawWethAsEth)
489     external
490     returns (
491       uint256[] memory tokenAmounts,
492       uint256 ethWithdrawn,
493       uint256 ethClaimed,
494       uint256 biosClaimed
495     )
496   {
497     (tokenAmounts, ethWithdrawn, ethClaimed, biosClaimed) =
            IUserPositions(
498       moduleMap.getModuleAddress(Modules.UserPositions)
499     ).withdrawAllAndClaim(msg.sender, tokens, withdrawWethAsEth);
500
501     if (ethWithdrawn > 0) {
502       IWeth9(
503         IIntegrationMap(moduleMap.getModuleAddress(Modules.
              IntegrationMap))
504           .getWethTokenAddress()
505       ).withdraw(ethWithdrawn);
506     }
507
508     if (ethWithdrawn + ethClaimed > 0) {
509
510       payable(msg.sender).transfer(ethWithdrawn + ethClaimed);
511     }
512
```

Listing 27: Kernel.sol (Lines 532)

```
526 function claimEthRewards() public returns (uint256 ethClaimed) {
527     ethClaimed = IUserPositions(
528       moduleMap.getModuleAddress(Modules.UserPositions)
529     ).claimEthRewards(msg.sender);
530
531
532     payable(msg.sender).transfer(ethClaimed);
533
534     emit ClaimEthRewards(msg.sender, ethClaimed);
535   }
536
```

Listing 28: YieldManager.sol (Lines 351)

```
318 function distributeEth() external override onlyController {
319     IIntegrationMap integrationMap = IIntegrationMap(
320       moduleMap.getModuleAddress(Modules.IntegrationMap)
321     );
322     address wethAddress = IIntegrationMap(integrationMap).
          getWethTokenAddress();
323
324     // First fill up gas wallet with ETH
325     ethToGasAccount();
326
327     uint256 wethToDistribute = IERC20MetadataUpgradeable(
          wethAddress).balanceOf(
328       address(this)
329     );
330
331     if (wethToDistribute > 0) {
332       uint256 biosBuyBackWethAmount = (wethToDistribute *
333         biosBuyBackEthWeight) / getEthWeightSum();
334       uint256 treasuryWethAmount = (wethToDistribute *
            treasuryEthWeight) /
335         getEthWeightSum();
336       uint256 protocolFeeWethAmount = (wethToDistribute *
337         protocolFeeEthWeight) / getEthWeightSum();
338       uint256 rewardsWethAmount = wethToDistribute -
339         biosBuyBackWethAmount -
340         treasuryWethAmount -
341         protocolFeeWethAmount;
342
343       // Send WETH to SushiSwap trader for BIOS buy back
344       IERC20MetadataUpgradeable(wethAddress).safeTransfer(
345         moduleMap.getModuleAddress(Modules.SushiSwapTrader),
346         biosBuyBackWethAmount
347       );
348
349       // Swap WETH for ETH and transfer to the treasury account
350       IWeth9(wethAddress).withdraw(treasuryWethAmount);
351       payable(treasuryAccount).transfer(treasuryWethAmount);
352
353       // Send ETH to protocol fee accrual rewards (BIOS stakers)
354       ethToProtocolFeeAccrual(protocolFeeWethAmount);
355
356       // Send ETH to token rewards
357       ethToRewards(rewardsWethAmount);
```

FINDINGS & TECH DETAILS

Listing 29: YieldManager.sol (Lines 382,390)

```
362 function ethToGasAccount() private {
363     address wethAddress = IIntegrationMap(
364       moduleMap.getModuleAddress(Modules.IntegrationMap)
365     ).getWethTokenAddress();
366     uint256 wethBalance = IERC20MetadataUpgradeable(wethAddress).
          balanceOf(
367       address(this)
368     );
369
370     if (wethBalance > 0) {
371       uint256 gasAccountActualEthBalance = gasAccount.balance;
372       if (gasAccountActualEthBalance < gasAccountTargetEthBalance)
              {
373         // Need to send ETH to gas account
374         uint256 ethAmountToGasAccount;
375         if (
376           wethBalance < gasAccountTargetEthBalance -
                gasAccountActualEthBalance
377         ) {
378           // Send all of WETH to gas wallet
379           ethAmountToGasAccount = wethBalance;
380           IWeth9(wethAddress).withdraw(ethAmountToGasAccount);
381
382           gasAccount.transfer(ethAmountToGasAccount);
383         } else {
384           // Send portion of WETH to gas wallet
385           ethAmountToGasAccount =
386             gasAccountTargetEthBalance -
387             gasAccountActualEthBalance;
388           IWeth9(wethAddress).withdraw(ethAmountToGasAccount);
389
390           gasAccount.transfer(ethAmountToGasAccount);
391         }
392       }
393     }
394   }
395
```

Listing 30: YieldManager.sol (Lines 449)

```
416 function ethToRewards(uint256 ethRewardsAmount) private {
417     uint256 processedWethByTokenSum = getProcessedWethSum();
418     require(
419       processedWethByTokenSum > 0,
420       "YieldManager::ethToRewards: No processed WETH to distribute
            "
421     );
422
423     IIntegrationMap integrationMap = IIntegrationMap(
424       moduleMap.getModuleAddress(Modules.IntegrationMap)
425     );
426     address wethAddress = integrationMap.getWethTokenAddress();
427     uint256 tokenCount = integrationMap.getTokenAddressesLength();
428
429     for (uint256 tokenId; tokenId < tokenCount; tokenId++) {
430       address tokenAddress = integrationMap.getTokenAddress(
            tokenId);
431
432       if (processedWethByToken[tokenAddress] > 0) {
433         IEtherRewards(moduleMap.getModuleAddress(Modules.
              EtherRewards))
434           .increaseEthRewards(
435             tokenAddress,
436             (ethRewardsAmount * processedWethByToken[tokenAddress
                  ]) /
437               processedWethByTokenSum
438           );
439
440         processedWethByToken[tokenAddress] = 0;
441       }
442     }
443
444     lastEthRewardsAmount = ethRewardsAmount;
445
446     IWeth9(wethAddress).withdraw(ethRewardsAmount);
447
448
449     payable(moduleMap.getModuleAddress(Modules.Kernel)).transfer(
450       ethRewardsAmount
451     );
452
453
```

Listing 31: YieldManager.sol (Lines 477,482)

```
456 function ethToProtocolFeeAccrual(uint256
      protocolFeeEthRewardsAmount)
457     private
458   {
459     IIntegrationMap integrationMap = IIntegrationMap(
460       moduleMap.getModuleAddress(Modules.IntegrationMap)
461     );
462     address biosAddress = integrationMap.getBiosTokenAddress();
463     address wethAddress = integrationMap.getWethTokenAddress();
464
465     if (
466       IStrategyMap(moduleMap.getModuleAddress(Modules.StrategyMap)
          )
467         .getTokenTotalBalance(biosAddress) > 0
468     ) {
469       // BIOS has been deposited, increase Ether rewards for BIOS
          depositors
470       IEtherRewards(moduleMap.getModuleAddress(Modules.
          EtherRewards))
471         .increaseEthRewards(biosAddress,
            protocolFeeEthRewardsAmount);
472
473       IWeth9(wethAddress).withdraw(protocolFeeEthRewardsAmount);
474
475
476
477       payable(moduleMap.getModuleAddress(Modules.Kernel)).transfer
          (
478         protocolFeeEthRewardsAmount
479       );
480     } else {
481       // No BIOS has been deposited, send WETH back to Kernel as
          reserves
482       IERC20MetadataUpgradeable(wethAddress).transfer(
483         moduleMap.getModuleAddress(Modules.Kernel),
484         protocolFeeEthRewardsAmount
485       );
486
487     }
488   }
```

Listing 32: YearnIntegration.sol (Lines 122)

```
101 function deploy() external override onlyController {
102     IIntegrationMap integrationMap = IIntegrationMap(
103       moduleMap.getModuleAddress(Modules.IntegrationMap)
104     );
105     uint256 tokenCount = integrationMap.getTokenAddressesLength();
106
107     for (uint256 tokenId = 0; tokenId < tokenCount; tokenId++) {
108       IERC20MetadataUpgradeable token = IERC20MetadataUpgradeable(
109         integrationMap.getTokenAddress(tokenId)
110       );
111       uint256 tokenAmount = token.balanceOf(address(this));
112       address vaultAddress = getVaultAddress(address(token));
113
114       // Check if a vault for this token exists
115       if (vaultAddress != address(0)) {
116         if (token.allowance(address(this), vaultAddress) == 0) {
117           token.safeApprove(vaultAddress, type(uint256).max);
118         }
119
120         if (tokenAmount > 0) {
121           try
122             IYearnVault(vaultAddress).deposit(tokenAmount, address
                    (this))
123
124           {} catch {}
125         }
126       }
127     }
128   }
```

Risk Level:

Likelihood - 3
Impact - 2

Recommendation:

Add return value check to avoid unexpected crash of the contract. Return value check will help in handling the exceptions better way.

Remediation Plan:

ACKNOWLEDGED: 0x_nodes acknowledged the issue and they will fix it in a future release.

3.8 (HAL-08) MISSING ZERO ADDRESS CHECK - LOW

Description:

Lack of zero address validation has been found at many instances in YieldManager.sol, UniswapIntegration.sol, when assigning user supplied address values to state variables directly.

Code Location:

Listing 33: IntegrationMap.sol (Lines 64)

```
63 function _addToken(
64     address tokenAddress,
65     bool acceptingDeposits,
66     bool acceptingWithdrawals,
67     uint256 biosRewardWeight,
68     uint256 reserveRatioNumerator
69   ) internal {
70     require(
71       !tokens[tokenAddress].added,
72       "IntegrationMap::addToken: Token already added"
73     );
74     require(
75       reserveRatioNumerator <= RESERVE_RATIO_DENOMINATOR,
76       "IntegrationMap::addToken: reserveRatioNumerator must be
          less than or equal to reserve ratio denominator"
77     );
78
79
80
81     tokens[tokenAddress].id = tokenAddresses.length;
82     tokens[tokenAddress].added = true;
83     tokens[tokenAddress].acceptingDeposits = acceptingDeposits;
84     tokens[tokenAddress].acceptingWithdrawals =
          acceptingWithdrawals;
85     tokens[tokenAddress].biosRewardWeight = biosRewardWeight;
86     tokens[tokenAddress].reserveRatioNumerator =
          reserveRatioNumerator;
87     tokenAddresses.push(tokenAddress);
```

```
88   }
```

Listing 34: Kernel.sol (Lines 169)

```
169 function addIntegration(address contractAddress, string memory
      name)
170     external
171     onlyRole(MANAGER_ROLE)
172   {
173
174     IIntegrationMap(moduleMap.getModuleAddress(Modules.
          IntegrationMap))
175       .addIntegration(contractAddress, name);
176
177     emit IntegrationAdded(contractAddress, name);
178   }
```

Listing 35: YieldManager.sol (Lines 109)

```
101 function updateGasAccount(address payable gasAccount_)
102     external
103     override
104     onlyController
105   {
106
107
108
109     gasAccount = gasAccount_;
110   }
```

Listing 36: YieldManager.sol (Lines 120)

```
113 function updateTreasuryAccount(address payable treasuryAccount_)
114     external
115     override
116     onlyController
117   {
118
119
120     treasuryAccount = treasuryAccount_;
121   }
```

Listing 37: DynamicRangeOrdersIntegration.sol (Lines 74)

```
73 function setDynamicRangeOrdersIntegrationDeployer(
74     address dynamicRangeOrdersIntegrationDeployerAddress
75   ) external override onlyOwner {
76
77     dynamicRangeOrdersIntegrationDeployer =
           IDynamicRangeOrdersIntegrationDeployer(
78       dynamicRangeOrdersIntegrationDeployerAddress
79     );
80   }
```

Listing 38: DynamicRangeOrdersIntegration.sol (Lines 83)

```
83 function setBaseStablecoin(address baseStablecoinAddress_)
84     external
85     override
86     onlyManager
87   {
88
89     require(
90       IIntegrationMap(moduleMap.getModuleAddress(Modules.
           IntegrationMap))
91         .getIsTokenAdded(baseStablecoinAddress_),
92       "DROIntegration::setBaseStablecoin: Token has not been added
             to IntegrationMap"
93     );
94     require(
95       baseStablecoinAddress_ != baseStablecoinAddress,
96       "DROIntegration::setBaseStablecoin: Address is already the
           base stablecoin"
97     );
98     baseStablecoinAddress = baseStablecoinAddress_;
99   }
```

- DynamicRangeOrdersIntegration.sol
  - deposit -> tokenAddress
  - withdraw -> tokenAddress
- AaveIntegration.sol
  - deposit -> tokenAddress

    - withdraw -> tokenAddress
- SushiSwapIntegration.sol
    - deposit -> tokenAddress
    - withdraw -> tokenAddress
- UniswapIntegration.sol
    - deposit -> tokenAddress
    - withdraw -> tokenAddress
- YearnIntegration.sol
    - deposit -> tokenAddress
    - withdraw -> tokenAddress

Recommendation:

Add proper address validation when every state variable assignment done from user supplied input.

Remediation Plan:

ACKNOWLEDGED: 0x_nodes acknowledged the issue and they will fix it in a future release.

3.9 (HAL-09) MISSING ZERO VALUE CHECK - LOW

Description:

There are functions within the project that should have a zero value check.

Code Location:

Listing 39: YieldManager.sol (Lines 81)

```
75 function updateGasAccountTargetEthBalance(uint256
       gasAccountTargetEthBalance_)
76     external
77     override
78     onlyController
79   {
80
81     gasAccountTargetEthBalance = gasAccountTargetEthBalance_;
82   }
```

Listing 40: YieldManager.sol (Lines 212)

```
206 function _calculateReserveAmount(
207     uint256 amount,
208     uint256 numerator,
209     uint256 denominator
210   ) internal pure returns (uint256) {
211
212     return (amount * numerator) / denominator;
213   }
```

- Also possible division by 0.

Listing 41: UniswapTrader.sol (Lines 213)

```
209 function updatePoolSlippageNumerator(
210     address tokenA,
211     address tokenB,
212     uint256 poolIndex,
213     uint24 slippageNumerator
214   ) external override onlyManager {
215
216
217     require(
218       slippageNumerator <= SLIPPAGE_DENOMINATOR,
219       "UniswapTrader:updatePoolSlippageNumerator: Slippage
            numerator must not be greater than slippage denominator"
220     );
221     (address token0, address token1) = getTokensSorted(tokenA,
          tokenB);
222     require(
223       pools[token0][token1][poolIndex].slippageNumerator !=
            slippageNumerator,
224       "UniswapTrader:updatePoolSlippageNumerator: Slippage
            numerator must be updated to a new number"
225     );
226     require(
227       pools[token0][token1].length > poolIndex,
228       "UniswapTrader:updatePoolSlippageNumerator: Pool does not
            exist"
229     );
230
231     pools[token0][token1][poolIndex].slippageNumerator =
          slippageNumerator;
232
233     emit UniswapPoolSlippageNumeratorUpdated(
234       token0,
235       token1,
236       poolIndex,
237       slippageNumerator
238     );
239   }
```

Listing 42: IntegrationMap.sol (Lines 68)

```
63 function _addToken(
64     address tokenAddress,
65     bool acceptingDeposits,
66     bool acceptingWithdrawals,
67     uint256 biosRewardWeight,
68     uint256 reserveRatioNumerator
69   ) internal {
70     require(
71       !tokens[tokenAddress].added,
72       "IntegrationMap::addToken: Token already added"
73     );
74     require(
75       reserveRatioNumerator <= RESERVE_RATIO_DENOMINATOR,
76       "IntegrationMap::addToken: reserveRatioNumerator must be
           less than or equal to reserve ratio denominator"
77     );
78
79
80
81     tokens[tokenAddress].id = tokenAddresses.length;
82     tokens[tokenAddress].added = true;
83     tokens[tokenAddress].acceptingDeposits = acceptingDeposits;
84     tokens[tokenAddress].acceptingWithdrawals =
         acceptingWithdrawals;
85     tokens[tokenAddress].biosRewardWeight = biosRewardWeight;
86     tokens[tokenAddress].reserveRatioNumerator =
         reserveRatioNumerator;
87     tokenAddresses.push(tokenAddress);
88   }
```

Listing 43: UniswapIntegration.sol (Lines 206)

```
198 function deposit(address tokenAddress, uint256 amount)
199     external
200     override
201     onlyController
202   {
203
204
205
206     balances[tokenAddress] += amount;
207   }
```

Listing 44: DynamicRangeOrdersIntegration.sol (Lines 168)

```
158 function _calculateReserveAmount(
159     uint256 amount,
160     uint256 numerator,
161     uint256 denominator
162   ) internal pure returns (uint256) {
163
164     return (amount * numerator) / denominator;
165   }
```

Listing 45: Kernel.sol (Lines 382)

```
380 function updateTokenReserveRatioNumerator(
381     address tokenAddress,
382     uint256 reserveRatioNumerator,
383     bool rebalance
384   ) external onlyRole(MANAGER_ROLE) {
385
386
387     IIntegrationMap(moduleMap.getModuleAddress(Modules.
          IntegrationMap))
388       .updateTokenReserveRatioNumerator(tokenAddress,
            reserveRatioNumerator);
389
390     if (rebalance) {
391       IYieldManager(moduleMap.getModuleAddress(Modules.
            YieldManager))
392         .rebalance();
393     }
394
395     emit TokenReserveRatioNumeratorUpdated(
396       tokenAddress,
397       reserveRatioNumerator,
398       rebalance
399     );
400   }
```

- DynamicRangeOrdersIntegration.sol
  - deposit -> amount
  - withdraw -> amount

- AaveIntegration.sol
  - deposit -> amount
  - withdraw -> amount
- SushiSwapIntegration.sol
  - deposit -> amount
  - withdraw -> amount
- UniswapIntegration.sol
  - deposit -> amount
  - withdraw -> amount
- YearnIntegration.sol
  - deposit -> amount
  - withdraw -> amount

Risk Level:

Likelihood - 3
Impact - 2

Recommendation:

It is recommended to add a zero value check to ensure that those variables were not set to 0.

Remediation Plan:

ACKNOWLEDGED: 0x_nodes acknowledged the issue and they will fix it in a future release.

3.10 (HAL-10) REDUNDANT CODE - LOW

Description:

The project contains code that does not needed to execute the logic of the smart contract. In Controlled.sol there is an address array called controllers which is not used in neither initialization to push controllers into thhe array nor in onlyController modifier. onlyController modifier depends on a mapping called _controllers that maps boolean to the address of the controller to denote whether the address is a controller.

Code Location:

Listing 46: Controlled.sol (Lines 11,13,54)

```
 8 abstract contract Controlled is Initializable, ModuleMapConsumer {
 9   // controller address => is a controller
10       mapping(address => bool) internal _controllers;
11   address[] public controllers;
12
13   function __Controlled_init(address[] memory controllers_,
       address moduleMap_)
14     public
15     initializer
16   {
17     for (uint256 i; i < controllers_.length; i++) {
18       _controllers[controllers_[i]] = true;
19     }
20     __ModuleMapConsumer_init(moduleMap_);
21   }
22
23   function addController(address controller) external onlyOwner {
24     _controllers[controller] = true;
25     bool added;
26     for (uint256 i; i < controllers.length; i++) {
27       if (controller == controllers[i]) {
28         added = true;
29       }
30     }
31     if (!added) {
32       controllers.push(controller);
```

```
33      }
34    }
35
36    modifier onlyOwner() {
37      require(
38        IKernel(moduleMap.getModuleAddress(Modules.Kernel)).isOwner(
              msg.sender),
39        "Controlled::onlyOwner: Caller is not owner"
40      );
41      _;
42    }
43
44    modifier onlyManager() {
45      require(
46        IKernel(moduleMap.getModuleAddress(Modules.Kernel)).
              isManager(msg.sender),
47        "Controlled::onlyManager: Caller is not manager"
48      );
49      _;
50    }
51
52    modifier onlyController() {
53      require(
54        _controllers[msg.sender],
55        "Controlled::onlyController: Caller is not controller"
56      );
57      _;
58    }
59  }
```

Risk Level:

Likelihood - 2
Impact - 2

Recommendation:

It is recommended to remove redundant controllers array or include it into the main logic of the contract.

Remediation Plan:

ACKNOWLEDGED: 0x_nodes acknowledged the issue and they will fix it in a future release.

3.11 (HAL-11) REDUNDANT INITIALIZATION - LOW

Description:

The project contains many instances of redundant initialization. There are many instances of ModuleMapConsumer through __ModuleMapConsumer_init (moduleMap_); in initialize function. It becomes redundant in cases when __Controlled_init(controllers_, moduleMap_); is also initialized inside of the intialize function. That's because __Controlled_init(controllers_, moduleMap_); is already contains __ModuleMapConsumer_init (moduleMap_); inside of it so there is no reason to additionally call __ModuleMapConsumer_init(moduleMap_);. Also, there is an instance of redundant initialization in Kernel.sol of __AccessControl_init();. This initializer calls other initializer functions that are empty. Therefore, keeping it won't make any difference

Code Location:

Listing 47: Kernel.sol (Lines 129)

```
123   function initialize(
124     address admin_,
125     address owner_,
126     address moduleMap_
127   ) external initializer {
128     __ModuleMapConsumer_init(moduleMap_);
129     __AccessControl_init();
130
131     // make the "admin_" address the default admin role
132     _setupRole(DEFAULT_ADMIN_ROLE, admin_);
133
134     // make the "owner_" address the owner of the system
135     _setupRole(OWNER_ROLE, owner_);
136
137     // give the "owner_" address the manager role, too
138     _setupRole(MANAGER_ROLE, owner_);
139
140     // owners are admins of managers
```

```
141     _setRoleAdmin(MANAGER_ROLE, OWNER_ROLE);
142
143     initializationTimestamp = block.timestamp;
144   }
```

Listing 48: Controlled.sol (Lines 20)

```
13    function __Controlled_init(address[] memory controllers_,
          address moduleMap_)
14      public
15      initializer
16    {
17      for (uint256 i; i < controllers_.length; i++) {
18        _controllers[controllers_[i]] = true;
19      }
20      __ModuleMapConsumer_init(moduleMap_);
21    }
```

Listing 49: BiosRewards.sol (Lines 34)

```
29    function initialize(address[] memory controllers_, address
          moduleMap_)
30      public
31      initializer
32    {
33      __Controlled_init(controllers_, moduleMap_);
34      __ModuleMapConsumer_init(moduleMap_);
35    }
```

Listing 50: EtherRewards.sol (Lines 23)

```
18    function initialize(address[] memory controllers_, address
          moduleMap_)
19      public
20      initializer
21    {
22      __Controlled_init(controllers_, moduleMap_);
23      __ModuleMapConsumer_init(moduleMap_);
24    }
```

Listing 51: SushiSwapTrader.sol (Lines 63)

```
51 function initialize(
52     address[] memory controllers_,
53     address moduleMap_,
54     address factoryAddress_,
55     address swapRouterAddress_,
56     uint24 slippageNumerator_
57   ) public initializer {
58     require(
59       slippageNumerator <= SLIPPAGE_DENOMINATOR,
60       "SushiSwapTrader::initialize: Slippage Numerator must be
           less than or equal to slippage denominator"
61     );
62     __Controlled_init(controllers_, moduleMap_);
63     __ModuleMapConsumer_init(moduleMap_);
64     factoryAddress = factoryAddress_;
65     swapRouterAddress = swapRouterAddress_;
66     slippageNumerator = slippageNumerator_;
67   }
```

Listing 52: UniswapTrader.sol (Lines 78)

```
71 function initialize(
72     address[] memory controllers_,
73     address moduleMap_,
74     address factoryAddress_,
75     address swapRouterAddress_
76   ) public initializer {
77     __Controlled_init(controllers_, moduleMap_);
78     __ModuleMapConsumer_init(moduleMap_);
79     factoryAddress = factoryAddress_;
80     swapRouterAddress = swapRouterAddress_;
81   }
```

Listing 53: UserPositions.sol (Lines 45)

```
39 function initialize(
40     address[] memory controllers_,
41     address moduleMap_,
42     uint32 biosRewardsDuration_
43   ) public initializer {
44     __Controlled_init(controllers_, moduleMap_);
45     __ModuleMapConsumer_init(moduleMap_);
46     _biosRewardsDuration = biosRewardsDuration_;
47   }
```

Listing 54: YieldManager.sol (Lines 64)

```
52 function initialize(
53     address[] memory controllers_,
54     address moduleMap_,
55     uint256 gasAccountTargetEthBalance_,
56     uint32 biosBuyBackEthWeight_,
57     uint32 treasuryEthWeight_,
58     uint32 protocolFeeEthWeight_,
59     uint32 rewardsEthWeight_,
60     address payable gasAccount_,
61     address payable treasuryAccount_
62   ) public initializer {
63     __Controlled_init(controllers_, moduleMap_);
64     __ModuleMapConsumer_init(moduleMap_);
65     gasAccountTargetEthBalance = gasAccountTargetEthBalance_;
66     biosBuyBackEthWeight = biosBuyBackEthWeight_;
67     treasuryEthWeight = treasuryEthWeight_;
68     protocolFeeEthWeight = protocolFeeEthWeight_;
69     rewardsEthWeight = rewardsEthWeight_;
70     gasAccount = gasAccount_;
71     treasuryAccount = treasuryAccount_;
72   }
```

Listing 55: AaveIntegration.sol (Lines 35)

```
29 function initialize(
30     address[] memory controllers_,
31     address moduleMap_,
32     address lendingPoolAddress_
33   ) public initializer {
34     __Controlled_init(controllers_, moduleMap_);
35     __ModuleMapConsumer_init(moduleMap_);
36     lendingPoolAddress = lendingPoolAddress_;
37   }
```

Listing 56: DynamicRangeOrdersIntegration.sol (Lines 67)

```
60 function initialize(
61     address[] memory controllers_,
62     address moduleMap_,
63     address factoryAddress_,
64     address positionManagerAddress_
65   ) public initializer {
66     __Controlled_init(controllers_, moduleMap_);
67     __ModuleMapConsumer_init(moduleMap_);
68     factoryAddress = factoryAddress_;
69     positionManagerAddress = positionManagerAddress_;
70   }
```

Listing 57: DynamicRangeOrdersIntegrationDeployer.sol (Lines 49)

```
41 function initialize(
42     address[] memory controllers_,
43     address moduleMap_,
44     address factoryAddress_,
45     address positionManagerAddress_,
46     address dynamicRangeOrdersIntegrationAddress_
47   ) public initializer {
48     __Controlled_init(controllers_, moduleMap_);
49     __ModuleMapConsumer_init(moduleMap_);
50     factoryAddress = factoryAddress_;
51     positionManagerAddress = positionManagerAddress_;
52     dynamicRangeOrdersIntegration = IDynamicRangeOrdersIntegration
           (
53       dynamicRangeOrdersIntegrationAddress_
54     );
55   }
```

Listing 58: UniswapIntegration.sol (Lines 77)

```
70 function initialize(
71     address[] memory controllers_,
72     address moduleMap_,
73     address factoryAddress_,
74     address positionManagerAddress_
75   ) public initializer {
76     __Controlled_init(controllers_, moduleMap_);
77     __ModuleMapConsumer_init(moduleMap_);
78     factoryAddress = factoryAddress_;
79     positionManagerAddress = positionManagerAddress_;
80   }
```

Listing 59: UniswapIntegrationDeployer.sol (Lines 49)

```
41 function initialize(
42     address[] memory controllers_,
43     address moduleMap_,
44     address factoryAddress_,
45     address positionManagerAddress_,
46     address uniswapIntegrationAddress_
47   ) public initializer {
48     __Controlled_init(controllers_, moduleMap_);
49     __ModuleMapConsumer_init(moduleMap_);
50     factoryAddress = factoryAddress_;
51     positionManagerAddress = positionManagerAddress_;
52     uniswapIntegration = IUniswapIntegration(
           uniswapIntegrationAddress_);
53   }
```

Listing 60: YearnIntegration.sol (Lines 35)

```
29   function initialize(
30     address[] memory controllers_,
31     address moduleMap_,
32     address yearnRegistryAddress_
33   ) public initializer {
34     __Controlled_init(controllers_, moduleMap_);
35     __ModuleMapConsumer_init(moduleMap_);
36     yearnRegistryAddress = yearnRegistryAddress_;
37   }
```

Risk Level:

Likelihood - 2
Impact - 2

Recommendation:

Consider removing redundant `__ModuleMapConsumer_init(moduleMap_);` from `initialize` functions that have `__Controlled_init(controllers_, moduleMap_);`. As another solution, remove `__ModuleMapConsumer_init(moduleMap_);` from `Controlled.sol` and keep `__ModuleMapConsumer_init(`

moduleMap_); in all initializers. Also, remove __AccessControl_init(); from the initializer in Kernel.sol.

Remediation Plan:

ACKNOWLEDGED: 0x_nodes acknowledged the issue and they will fix it in a future release.

3.12 (HAL-12) INCORRECT INITIALIZATION - LOW

Description:

In ModuleMap.sol the project uses __Ownable_init(); inside of initializer to initialize the OwnableUpgradable contract. However, there is a call to __Context_init_unchained(); within __Ownable_init();which is empty since it was automatically generated by OpenZepplin transpiler.

Code Location:

Listing 61: ModuleMap.sol (Lines 12)

```
11   function initialize() public initializer {
12     __Ownable_init();
13   }
```

Risk Level:

Likelihood - 2
Impact - 2

Recommendation:

Instead of initializing OwnableUpgradable through __Ownable_init(); that refers to empty Context initializer, initialize OwnableUpgradable through __Ownable_init_unchained() directly.

Listing 62: ModuleMap.sol (Lines 12)

```
11   function initialize() public initializer {
12     __Ownable_init_unchained();
13   }
```

Remediation Plan:

SOLVED: 0x_nodes solved the issue by utilizing correct initializer.

3.13 (HAL-13) REDUNDANT ARITHMETICAL OPERATIONS - LOW

Description:

In the SushiSwapIntegration.sol, there are instances when a variable is multiplied by 1. It is redundant since it does not change the underlying value.

Code Location:

Listing 63: SushiSwapIntegration.sol (Lines 186)

```
169  function _deploy(uint256 amount, uint256 pid) internal {
170          ISushiSwapIntegration.InnerPool memory innerPoolInfo =
                 getPoolInfo(pid);
171          require(innerPoolInfo.added, "SushiSwapIntegration::
                 deposit to sushiSwap yield farm: pool is not configured
                 ");
172
173          (uint256 priceWithMultiplier, uint256
                 decimalsSubtractionExponent) = getPriceWithMultiplier(
                 getTokenToAdd(innerPoolInfo.tokenPair.token0,
                 innerPoolInfo.tokenPair.token1), address(innerPoolInfo.
                 poolInfo.lpToken), innerPoolInfo.tokenPair.token0);
174          (uint256 amountTokenDesired, uint256 amountTokenMin,
                 uint256 amountWeiMin) =
                 calculateAmountOfTokenToAddLiquidityETH(amount / 2,
                 priceWithMultiplier);
175          uint256 tokensReceived;
176
177          if(IERC20MetadataUpgradeable(IIntegrationMap(moduleMap.
                 getModuleAddress(Modules.IntegrationMap)).
                 getWethTokenAddress()).balanceOf(address(this)) <
                 amount) {
178              return;
179          }
180          IWeth9(IIntegrationMap(moduleMap.getModuleAddress(Modules.
                 IntegrationMap)).getWethTokenAddress()).withdraw(amount
                 );
181
```

```
182         if (IERC20MetadataUpgradeable(getTokenToAdd(innerPoolInfo.
                tokenPair.token0, innerPoolInfo.tokenPair.token1)).
                balanceOf(moduleMap.getModuleAddress(Modules.Kernel)) <
                 amountTokenDesired / decimalsSubtractionExponent) {
183
184             uint[] memory amounts = swapExactETHForTokens(
185                 amount / 2,
186                 ((amountTokenDesired - (amountTokenDesired * 1 /
                        100)) / decimalsSubtractionExponent),
187                 wethAddress,
188                 getTokenToAdd(innerPoolInfo.tokenPair.token0,
                        innerPoolInfo.tokenPair.token1),
189                 address(this)
190             );
191             ...
192             }
```

Listing 64: SushiSwapIntegration.sol (Lines 263,264)

```
257     function calculateAmountOfTokenToAddLiquidityETH(
258         uint256 amountWei,
259         uint256 priceWithMultiplier
260     ) public pure override returns (uint256, uint256, uint256) {
261
262         uint256 amountTokenDesired = amountWei *
                priceWithMultiplier / MULTIPLIER;
263         uint256 amountTokenMin = amountTokenDesired - (
                amountTokenDesired * 1) / 100; // 1% tolerance level
264         uint256 amountWeiMin = amountWei - (amountWei * 1) / 100;
                // 1% tolerance level
265
266         return (amountTokenDesired, amountTokenMin, amountWeiMin);
267     }
```

Risk Level:

Likelihood - 2

Impact - 2

Recommendation:

It is recommended to remove multiplication by 1 since it does not change the value.

Remediation Plan:

ACKNOWLEDGED: 0x_nodes acknowledged the issue and they will fix it in a future release.

3.14 (HAL-14) INCORRECT EVENT ARGUMENTS - LOW

Description:

In deposit function within Kernel.sol emitted event does not take into account the actual value that was deposited to UserPositions.

Code Location:

Listing 65: Kernel.sol (Lines 452)

```
438   function deposit(address[] memory tokens, uint256[] memory
          amounts)
439     external
440     payable
441   {
442     if (msg.value > 0) {
443       // Convert ETH to WETH
444       address wethAddress = IIntegrationMap(
445         moduleMap.getModuleAddress(Modules.IntegrationMap)
446       ).getWethTokenAddress();
447       IWeth9(wethAddress).deposit{ value: msg.value }();
448     }
```

Risk Level:

Likelihood - 3
Impact - 1

Recommendation:

There are two possible solutions for this:

- Return actual amounts array from IUserPositions(moduleMap.getModuleAddress(Modules.UserPositions)).deposit() function and

put it inside of event. You can do it by creating an array with actual amounts within UserPositions.deposit, populating it, and returning it back.

- Emitting an event directly within UserPositions.deposit. You can create an array with actual amounts, populate it, and emit event directly within UserPositions contract.

Remediation Plan:

SOLVED: 0x_nodes solved the issue by creating an array of actual amounts and emitting it within UserPositions.sol

3.15 (HAL-15) FLOATING PRAGMA - LOW

Description:

The project contains many instances of floating pragma. Contracts should be deployed with the same compiler version and flags that they have been tested with thoroughly. Locking the pragma helps to ensure that contracts do not accidentally get deployed using, for example, either an outdated compiler version that might introduce bugs that affect the contract system negatively or a pragma version too new which has not been extensively tested.

Code Location:

Listing 66

```
contracts/core/YieldManager.sol:2:pragma solidity ^0.8.4;
contracts/core/Controlled.sol:2:pragma solidity ^0.8.4;
contracts/core/UserPositions.sol:2:pragma solidity ^0.8.4;
contracts/core/BiosRewards.sol:2:pragma solidity ^0.8.4;
contracts/core/UniswapTrader.sol:2:pragma solidity ^0.8.4;
contracts/core/StrategyMap.sol:2:pragma solidity ^0.8.4;
contracts/core/StrategyManager.sol:2:pragma solidity ^0.8.4;
contracts/core/SushiSwapTrader.sol:2:pragma solidity ^0.8.4;
contracts/core/EtherRewards.sol:2:pragma solidity ^0.8.4;
contracts/core/ModuleMap.sol:2:pragma solidity ^0.8.4;
contracts/core/Kernel.sol:2:pragma solidity ^0.8.4;
contracts/core/IntegrationMap.sol:2:pragma solidity ^0.8.4;
contracts/core/ModuleMapConsumer.sol:2:pragma solidity ^0.8.4;
contracts/yield-integrations/AaveIntegration.sol:2:pragma solidity
    ^0.8.4;
contracts/yield-integrations/YearnIntegration.sol:2:pragma
    solidity ^0.8.4;
contracts/yield-integrations/UniswapIntegrationDeployer.sol:2:
    pragma solidity ^0.8.4;
contracts/yield-integrations/UniswapIntegration.sol:2:pragma
    solidity ^0.8.4;
contracts/interfaces/IUniswapSwapRouter.sol:2:pragma solidity
    ^0.8.4;
```

```
19 contracts/interfaces/IYieldManager.sol:2:pragma solidity ^0.8.4;
20 contracts/interfaces/IUniswapTrader.sol:2:pragma solidity ^0.8.4;
21 contracts/interfaces/IUserPositions.sol:2:pragma solidity ^0.8.4;
22 contracts/interfaces/IUniswapIntegration.sol:2:pragma solidity
       ^0.8.4;
23 contracts/interfaces/IYearnRegistry.sol:2:pragma solidity ^0.8.4;
24 contracts/interfaces/IWeth9.sol:2:pragma solidity ^0.8.4;
25 contracts/interfaces/IBiosRewards.sol:2:pragma solidity ^0.8.4;
26 contracts/interfaces/ISushiSwapRouter.sol:2:pragma solidity
       ^0.8.4;
27 contracts/interfaces/ISushiSwapPair.sol:2:pragma solidity ^0.8.4;
28 contracts/interfaces/IYearnVault.sol:2:pragma solidity ^0.8.4;
29 contracts/interfaces/IUniswapPool.sol:2:pragma solidity ^0.8.4;
30 contracts/interfaces/IStrategyManager.sol:2:pragma solidity
       ^0.8.4;
31 contracts/interfaces/IUniswapIntegrationDeployer.sol:2:pragma
       solidity ^0.8.4;
32 contracts/interfaces/ISushiSwapTrader.sol:2:pragma solidity
       ^0.8.4;
33 contracts/interfaces/IAaveLendingPool.sol:2:pragma solidity
       ^0.8.4;
34 contracts/interfaces/IEtherRewards.sol:2:pragma solidity ^0.8.4;
35 contracts/interfaces/IUniswapPositionManager.sol:2:pragma solidity
        ^0.8.4;
36 contracts/interfaces/ISushiSwapFactory.sol:2:pragma solidity
       ^0.8.4;
37 contracts/interfaces/IUniswapFactory.sol:2:pragma solidity ^0.8.4;
38 contracts/interfaces/IKernel.sol:2:pragma solidity ^0.8.4;
39 contracts/interfaces/IModuleMap.sol:2:pragma solidity ^0.8.4;
40 contracts/interfaces/IIntegrationMap.sol:2:pragma solidity ^0.8.4;
41 contracts/interfaces/IIntegration.sol:2:pragma solidity ^0.8.4;
42 contracts/interfaces/IStrategyMap.sol:2:pragma solidity ^0.8.4;
```

Risk Level:

Likelihood - 3
Impact - 1

Recommendation:

Consider locking the pragma version with known bugs for the compiler version. When possible, do not use floating pragma in the final live deployment. Specifying a fixed compiler version ensures that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

Remediation Plan:

SOLVED: 0x_nodes team solved the issue by locking the pragma version.

3.16 (HAL-16) MISSING EVENT EMITTING - INFORMATIONAL

Description:

Functions that have important functionality in the YieldManager.sol, DynamicRangeOrdersIntegration.sol, SushiSwapIntegration.sol ,UniswapIntegration.sol, and YearnIntegration.sol contracts are missing event emitting. These functions should emit events.

Code Location:

- DynamicRangeOrdersIntegration.sol
  - deposit
  - withdraw
  - updateLiquidityPositionWeight
  - harvestYield
  - setBaseStablecoin
  - setDynamicRangeOrdersIntegrationDeployer
  - configureLiquidityPosition
  - deploy
- AaveIntegration.sol
  - deposit
  - withdraw
  - deploy
  - harvestYield
- SushiSwapIntegraion.sol
  - deposit
  - withdraw
  - addPool
  - updatePoolWeight
  - deploy
- UniswapIntegration.sol

    - deposit
    - withdraw
    - setUniswapIntegrationDeployer
    - setBaseStablecoin
    - configureLiquidityPosition
    - updateLiquidityPositionWeight
    - harvestYield
- YearnIntegration.sol
    - deposit
    - withdraw
    - harvestYield

Risk Level:

Likelihood - 2
Impact - 1

Recommendation:

Consider adding event emitting to mentioned functions

Remediation Plan:

ACKNOWLEDGED: 0x_nodes acknowledged the issue and they will fix it in a future release.

3.17 (HAL-17) PRAGMA TOO RECENT - INFORMATIONAL

Description:

The project uses one of the latest pragma version (0.8.4) which was released on June 22nd, 2021. The latest pragma version (0.8.8) was released in September 2021. Many pragma versions have been lately released, going from version 0.7.x to the recently released version 0.8.x. in just 6 months.

In the Solitidy Github repository, there is a json file where are all bugs finding in the different compiler versions. It should be noted that pragma 0.6.12 and 0.7.6 are widely used by Solidity developers and have been extensively tested in many security audits.

Code Location:

Listing 67

```
contracts/core/YieldManager.sol:2:pragma solidity ^0.8.4;
contracts/core/Controlled.sol:2:pragma solidity ^0.8.4;
contracts/core/UserPositions.sol:2:pragma solidity ^0.8.4;
contracts/core/BiosRewards.sol:2:pragma solidity ^0.8.4;
contracts/core/UniswapTrader.sol:2:pragma solidity ^0.8.4;
contracts/core/StrategyMap.sol:2:pragma solidity ^0.8.4;
contracts/core/StrategyManager.sol:2:pragma solidity ^0.8.4;
contracts/core/SushiSwapTrader.sol:2:pragma solidity ^0.8.4;
contracts/core/EtherRewards.sol:2:pragma solidity ^0.8.4;
contracts/core/ModuleMap.sol:2:pragma solidity ^0.8.4;
contracts/core/Kernel.sol:2:pragma solidity ^0.8.4;
contracts/core/IntegrationMap.sol:2:pragma solidity ^0.8.4;
contracts/core/ModuleMapConsumer.sol:2:pragma solidity ^0.8.4;
contracts/yield-integrations/AaveIntegration.sol:2:pragma solidity
    ^0.8.4;
contracts/yield-integrations/YearnIntegration.sol:2:pragma
    solidity ^0.8.4;
contracts/yield-integrations/UniswapIntegrationDeployer.sol:2:
    pragma solidity ^0.8.4;
```

```
17 contracts/yield-integrations/UniswapIntegration.sol:2:pragma
     solidity ^0.8.4;
18 contracts/interfaces/IUniswapSwapRouter.sol:2:pragma solidity
     ^0.8.4;
19 contracts/interfaces/IYieldManager.sol:2:pragma solidity ^0.8.4;
20 contracts/interfaces/IUniswapTrader.sol:2:pragma solidity ^0.8.4;
21 contracts/interfaces/IUserPositions.sol:2:pragma solidity ^0.8.4;
22 contracts/interfaces/IUniswapIntegration.sol:2:pragma solidity
     ^0.8.4;
23 contracts/interfaces/IYearnRegistry.sol:2:pragma solidity ^0.8.4;
24 contracts/interfaces/IWeth9.sol:2:pragma solidity ^0.8.4;
25 contracts/interfaces/IBiosRewards.sol:2:pragma solidity ^0.8.4;
26 contracts/interfaces/ISushiSwapRouter.sol:2:pragma solidity
     ^0.8.4;
27 contracts/interfaces/ISushiSwapPair.sol:2:pragma solidity ^0.8.4;
28 contracts/interfaces/IYearnVault.sol:2:pragma solidity ^0.8.4;
29 contracts/interfaces/IUniswapPool.sol:2:pragma solidity ^0.8.4;
30 contracts/interfaces/IStrategyManager.sol:2:pragma solidity
     ^0.8.4;
31 contracts/interfaces/IUniswapIntegrationDeployer.sol:2:pragma
     solidity ^0.8.4;
32 contracts/interfaces/ISushiSwapTrader.sol:2:pragma solidity
     ^0.8.4;
33 contracts/interfaces/IAaveLendingPool.sol:2:pragma solidity
     ^0.8.4;
34 contracts/interfaces/IEtherRewards.sol:2:pragma solidity ^0.8.4;
35 contracts/interfaces/IUniswapPositionManager.sol:2:pragma solidity
      ^0.8.4;
36 contracts/interfaces/ISushiSwapFactory.sol:2:pragma solidity
     ^0.8.4;
37 contracts/interfaces/IUniswapFactory.sol:2:pragma solidity ^0.8.4;
38 contracts/interfaces/IKernel.sol:2:pragma solidity ^0.8.4;
39 contracts/interfaces/IModuleMap.sol:2:pragma solidity ^0.8.4;
40 contracts/interfaces/IIntegrationMap.sol:2:pragma solidity ^0.8.4;
41 contracts/interfaces/IIntegration.sol:2:pragma solidity ^0.8.4;
42 contracts/interfaces/IStrategyMap.sol:2:pragma solidity ^0.8.4;
```

Risk Level:

Likelihood - 2
Impact - 1

Recommendation:

If possible, consider using the latest stable pragma version that has been thoroughly tested to prevent potential undiscovered vulnerabilities such as pragma between 0.6.12 - 0.7.6.

References:

- Solidity Releases
- Solidity Bugs By Version

Remediation Plan:

ACKNOWLEDGED: 0x_nodes considers usage of recent pragma version safe since the use of solidity 0.8.4 was a deliberate design choice to gain access to built in safe math, and abicoder v2 (so that structs can be passed as function parameters). While the team considered the use of 0.7.6, it was felt that access to the additional features, and the latest bugfixes represented a better choice for the project.

3.18 (HAL-18) AMBIGUOUS DESIGN - INFORMATIONAL

Description:

There are some design decision within the project that are hard to understand, which in return decrease overall readability of the project.

Code Locaiton:

increaseRewards and decreaseRewards functions logically are exactly the same but they are separated under two separate functions.

Listing 68: BiosRewards.sol (Lines)

```
68   function increaseRewards(
69     address token,
70     address account,
71     uint256 amount
72   ) public override onlyController updateReward(token, account) {
73     require(amount > 0, "BiosRewards::increaseRewards: Cannot
         increase 0");
74   }
75
76   function decreaseRewards(
77     address token,
78     address account,
79     uint256 amount
80   ) public override onlyController updateReward(token, account) {
81     require(amount > 0, "BiosRewards::decreaseRewards: Cannot
         decrease 0");
82   }
83
```

Both rebalance and deploy call exactly the same _deploy function even though there is a separate _rebalance function.

Listing 69: YieldManager.sol (Lines 126,131,141)

```
125   function rebalance() external override onlyController {
126     _deploy();
127   }
128
129   /// @notice Deploys all tokens to all integrations according to
          configured weights
130   function deploy() external override onlyController {
131     _deploy();
132   }
133
134   function _deploy() internal {
135     bool shouldRedeploy = _rebalance();
136     if (shouldRedeploy) {
137       _rebalance();
138     }
139   }
140
141   function _rebalance() internal returns (bool redeploy) {
```

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

Consider either refactoring the code to decrease ambiguity or adding relevant comments to clarify design decisions.

Remediation Plan:

ACKNOWLEDGED: 0x_nodes acknowledged the issue and they will fix it in a future release.

3.19 (HAL-19) STATE VARIABLE VISIBILITY NOT SET - INFORMATIONAL

Description:

It is best practice to set the visibility of state variables explicitly.

Code Location:

Listing 70: DynamicRangeOrdersIntegration.sol (Lines 47,48,50)

```
45 mapping(bytes32 => LiquidityPosition) public liquidityPositions;
46   bytes32[] private liquidityPositionKeys;
47   mapping(bytes32 => uint256) liquidityPositionKeyIndexes;
48   mapping(bytes32 => uint256) lastRerangeTimestamp;
49   mapping(address => uint256) private balances;
50   IDynamicRangeOrdersIntegrationDeployer
         dynamicRangeOrdersIntegrationDeployer;
51
```

Listing 71: SushiSwapIntegration.sol (Lines 31,33,34,35,36,37)

```
26
27  using SafeERC20Upgradeable for IERC20MetadataUpgradeable;
28
29     uint256 private constant MULTIPLIER = 10 ** 18;
30
31     uint24 private constant SLIPPAGE_DENOMINATOR = 1_000_000;
32     uint24 slippageNumerator;
33
34     address factoryAddress;
35     address swapRouterAddress;
36     address masterChef;
37     address sushi;
38     address wethAddress;
39
```

Listing 72: UniswapIntegration.sol (Lines 45)

```
43  LiquidityPosition[] private liquidityPositions;
44   mapping(address => uint256) private balances;
45   IUniswapIntegrationDeployer uniswapIntegrationDeployer;
46
47   address private factoryAddress;
48   address private positionManagerAddress;
49   address private baseStablecoinAddress;
50
```

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

It is recommended to explicitly specify the visibility of all state variables.

Remediation Plan:

ACKNOWLEDGED: 0x_nodes acknowledged the issue and they will fix it in a future release.

3.20 (HAL-20) POSSIBLE MISUSE OF PUBLIC FUNCTIONS - INFORMATIONAL

Description:

In public functions, array arguments are immediately copied to memory, while external functions can read directly from `calldata`. Reading `calldata` is cheaper than memory allocation. Public functions need to write the arguments to memory because public functions may be called internally. Internal calls are passed internally by pointers to memory. Thus, the function expects its arguments being located in memory when the compiler generates the code for an internal function.
Also, methods do not necessarily have to be public if they are only called within the contract-in such case they should be marked `internal`.

Code Location:

- ModuleMap.sol
  - `setModuleAddress`
- SushiSwapTrader.sol
  - `getFactoryAddress`
  - `getSlippageNumerator`
  - `getSlippageDenominator`
- UniswapTrader.sol
  - `setPathFor`
  - `getTokensAndAmountsSorted`
  - `getSlippageDenominator`

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

Consider as much as possible declaring functions external instead of public. As for best practice, you should use external if you expect that the function will only be called externally and use public if you need to call the function internally. To sum up, all can access to public functions, external functions only can be accessed externally and internal functions can only be called within the contract.

Remediation Plan:

SOLVED: 0x_nodes solved the issue by declaring functions as external.

3.21 (HAL-21) DUPLICATE STATEMENTS - INFORMATIONAL

Description:

In harvestYield function within YieldManager.sol, tokenDesiredReserve - tokenActualReserve statement is duplicated.

Code Location:

Listing 73: YieldManager.sol (Lines 250,254)

```
216 function harvestYield() public override onlyController {
217     IIntegrationMap integrationMap = IIntegrationMap(
218       moduleMap.getModuleAddress(Modules.IntegrationMap)
219     );
220     uint256 tokenCount = integrationMap.getTokenAddressesLength();
221     uint256 integrationCount = integrationMap.
            getIntegrationAddressesLength();
222
223     for (
224       uint256 integrationId;
225       integrationId < integrationCount;
226       integrationId++
227     ) {
228       IIntegration(integrationMap.getIntegrationAddress(
              integrationId))
229         .harvestYield();
230     }
231
232     for (uint256 tokenId; tokenId < tokenCount; tokenId++) {
233       IERC20MetadataUpgradeable token = IERC20MetadataUpgradeable(
234         integrationMap.getTokenAddress(tokenId)
235       );
236
237       uint256 tokenDesiredReserve = getDesiredReserveTokenBalance(
238         address(token)
239       );
240       uint256 tokenActualReserve = getReserveTokenBalance(address(
              token));
241
```

```
242       uint256 harvestedTokenAmount = token.balanceOf(address(this)
              );
243
244
245
246
247       // Check if reserves need to be replenished
248       if (tokenDesiredReserve > tokenActualReserve) {
249         // Need to replenish reserves
250         if (tokenDesiredReserve - tokenActualReserve <=
                harvestedTokenAmount) {
251           // Need to send portion of harvested token to Kernel to
                replenish reserves
252           token.safeTransfer(
253             moduleMap.getModuleAddress(Modules.Kernel),
254             tokenDesiredReserve - tokenActualReserve
255           );
256         }
257       ...
```

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

It is recommended to extract these statements to a single variable to increase readability.
Possible Fix:

Listing 74: YieldManager.sol (Lines 2,7,11)

```
1
2   uint256 reserve = tokenDesiredReserve - tokenActualReserve;
3
4       // Check if reserves need to be replenished
5       if (tokenDesiredReserve > tokenActualReserve) {
6         // Need to replenish reserves
7         if (reserve <= harvestedTokenAmount) {
```

```
8           // Need to send portion of harvested token to Kernel to
                replenish reserves
9           token.safeTransfer(
10            moduleMap.getModuleAddress(Modules.Kernel),
11            reserve
12          );
13        }
14
```

Remediation plan:

SOLVED: 0x_nodes team solved this issue by refactoring the code.

AUTOMATED TESTING

4.1 STATIC ANALYSIS REPORT

Description:

Halborn used automated testing techniques to enhance coverage of certain areas of the scoped contract. Among the tools used was Slither, a Solidity static analysis framework. After Halborn verified all the contracts in the repository and was able to compile them correctly into their abi and binary formats. This tool can statically verify mathematical relationships between Solidity variables to detect invalid or inconsistent usage of the contracts' APIs across the entire code-base.

Results:

All the findings apart from HAL-04 and HAL-06 were either detected during manual code review or false positive.

```
Kernel.withdraw(address[],uint256[],bool) (contracts/core/Kernel.sol#443-481) sends eth to arbitrary user
	Dangerous calls:
	- address(msg.sender).transfer(ethWithdrawn) (contracts/core/Kernel.sol#477-478)
Kernel.withdrawAllAndClaim(address[],bool) (contracts/core/Kernel.sol#493-521) sends eth to arbitrary user
	Dangerous calls:
	- address(msg.sender).transfer(ethWithdrawn + ethClaimed) (contracts/core/Kernel.sol#511-515)
Kernel.claimEthRewards() (contracts/core/Kernel.sol#527-536) sends eth to arbitrary user
	Dangerous calls:
	- address(msg.sender).transfer(ethClaimed) (contracts/core/Kernel.sol#534-535)
YieldManager.distributeEth() (contracts/core/YieldManager.sol#318-359) sends eth to arbitrary user
	Dangerous calls:
	- address(treasuryAccount).transfer(treasuryWethAmount) (contracts/core/YieldManager.sol#351)
YieldManager.ethToGasAccount() (contracts/core/YieldManager.sol#362-394) sends eth to arbitrary user
	Dangerous calls:
	- gasAccount.transfer(ethAmountToGasAccount) (contracts/core/YieldManager.sol#382)
	- gasAccount.transfer(ethAmountToGasAccount) (contracts/core/YieldManager.sol#390)
YieldManager.ethToRewards(uint256) (contracts/core/YieldManager.sol#410-452) sends eth to arbitrary user
	Dangerous calls:
	- address(moduleMap.getModuleAddress(Modules.Kernel)).transfer(ethRewardsAmount) (contracts/core/YieldManager.sol#449-451)
YieldManager.ethToProtocolFeeAccrual(uint256) (contracts/core/YieldManager.sol#458-488) sends eth to arbitrary user
	Dangerous calls:
	- address(moduleMap.getModuleAddress(Modules.Kernel)).transfer(protocolFeeEthRewardsAmount) (contracts/core/YieldManager.sol#477-479)
SushiSwapIntegration._deploy(uint256,uint256) (contracts/yield-integrations/SushiSwapIntegration.sol#175-203) sends eth to arbitrary user
	Dangerous calls:
	- (liquidity) = ISushiSwapRouter(swapRouterAddress).addLiquidityETH{value: amount / 2}(getTokenToAdd(innerPoolInfo.tokenPair.token0,innerPoolInfo.tokenPair.token1),(amountTokenDesired / decimalsSubtractionExponent) - diff,(amountTokenMin / decimalsSubtractionExponent),amountWeiMin,address(this),block.timestam
p + 360) (contracts/yield-integrations/SushiSwapIntegration.sol#207-214)
	- IWeth(IIntegrationMap(moduleMap.getModuleAddress(Modules.IntegrationMap)).getWethTokenAddress()).deposit{value: ethBalance}() (contracts/yield-integrations/SushiSwapIntegration.sol#223)
SushiSwapIntegration.swapExactTHForTokens(uint256,uint256,address,address,address) (contracts/yield-integrations/SushiSwapIntegration.sol#364-377) sends eth to arbitrary user
	Dangerous calls:
	- ISushiSwapRouter(swapRouterAddress).swapExactETHForTokens{value: amountWei}(amountOutMin,path,to,deadline) (contracts/yield-integrations/SushiSwapIntegration.sol#376)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#functions-that-send-ether-to-arbitrary-destinations

Reentrancy in DynamicRangeOrdersIntegration._deploy() (contracts/yield-integrations/DynamicRangeOrdersIntegration.sol#269-337):
	External calls:
	- transferTokensToDynamicRangeOrdersIntegrationDeployer() (contracts/yield-integrations/DynamicRangeOrdersIntegration.sol#281)
		- returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/utils/SafeERC20Upgradeable.sol#71)
		- (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#139)
		- dynamicRangeOrdersIntegrationDeployer.tokenApprovals(address(token)) (contracts/yield-integrations/DynamicRangeOrdersIntegration.sol#382)
		- token.safeTransferFrom(address(dynamicRangeOrdersIntegrationDeployer),token.balanceOf(address(this))) (contracts/yield-integrations/DynamicRangeOrdersIntegration.sol#386-389)
	- dynamicRangeOrdersIntegrationDeployer.swapExcessTokensForBaseStablecoin(tokenDesiredAmounts) (contracts/yield-integrations/DynamicRangeOrdersIntegration.sol#326-328)
	- dynamicRangeOrdersIntegrationDeployer.swapExcessBaseStablecoinForTokens(tokenDesiredAmounts) (contracts/yield-integrations/DynamicRangeOrdersIntegration.sol#331-333)
	- increaseLiquidityPositions(liquidityPositionDesiredDeployAmounts) (contracts/yield-integrations/DynamicRangeOrdersIntegration.sol#336)
		- dynamicRangeOrdersIntegrationDeployer.increaseLiquidityPosition(liquidityPositions[liquidityPositionKeys[liquidityPositionIndex]].id,liquidityPositions[liquidityPositionKeys[liquidityPositionIndex]].token0,liquidityPositions[liquidityPositionKeys[liquidityPositionIndex]].token1,amount0Desired,amount
1Desired) (contracts/yield-integrations/DynamicRangeOrdersIntegration.sol#415-424)
		- (success,liquidityPositionId) = dynamicRangeOrdersIntegrationDeployer.mintLiquidityPosition(liquidityPositions[liquidityPositionKeys[liquidityPositionIndex]].token0,liquidityPositions[liquidityPositionKeys[liquidityPositionIndex]].token1,liquidityPositions[liquidityPositionKeys[liquidityPositionInde
x]].feeNumerator,liquidityPositions[liquidityPositionKeys[liquidityPositionIndex]].tickLower,liquidityPositions[liquidityPositionKeys[liquidityPositionIndex]].tickUpper,amount0Desired_scope_0,amount1Desired_scope_1) (contracts/yield-integrations/DynamicRangeOrdersIntegration.sol#437-455)
	External calls sending eth:
	- transferTokensToDynamicRangeOrdersIntegrationDeployer() (contracts/yield-integrations/DynamicRangeOrdersIntegration.sol#283)
		- (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#139)
	State variables written after the call(s):
	- increaseLiquidityPositions(liquidityPositionDesiredDeployAmounts) (contracts/yield-integrations/DynamicRangeOrdersIntegration.sol#336)
		- liquidityPositions[liquidityPositionKeys[liquidityPositionIndex]].minted = true (contracts/yield-integrations/DynamicRangeOrdersIntegration.sol#457-458)
		- liquidityPositions[liquidityPositionKeys[liquidityPositionIndex]].id = liquidityPositionId (contracts/yield-integrations/DynamicRangeOrdersIntegration.sol#459-460)

Reentrancy in UniswapIntegration._deploy() (contracts/yield-integrations/UniswapIntegration.sol#282-349):
	External calls:
	- transferTokensToUniswapIntegrationDeployer() (contracts/yield-integrations/UniswapIntegration.sol#294)
		- returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/utils/SafeERC20Upgradeable.sol#71)
		- (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#139)
		- uniswapIntegrationDeployer.tokenApprovals(address(token)) (contracts/yield-integrations/UniswapIntegration.sol#393)
		- token.safeTransferFrom(address(uniswapIntegrationDeployer),token.balanceOf(address(this))) (contracts/yield-integrations/UniswapIntegration.sol#396-399)
	- uniswapIntegrationDeployer.swapExcessTokensForBaseStablecoin(tokenDesiredAmounts) (contracts/yield-integrations/UniswapIntegration.sol#338-340)
	- uniswapIntegrationDeployer.swapExcessBaseStablecoinForTokens(tokenDesiredAmounts) (contracts/yield-integrations/UniswapIntegration.sol#343-345)
	- increaseLiquidityPositions(liquidityPositionDesiredDeployAmounts) (contracts/yield-integrations/UniswapIntegration.sol#346)
		- uniswapIntegrationDeployer.increaseLiquidityPosition(liquidityPositions[liquidityPositionKeys[liquidityPositionIndex]].id,liquidityPositions[liquidityPositionKeys[liquidityPositionIndex]].token0,liquidityPositions[liquidityPositionKeys[liquidityPositionIndex]].token1,amount0Desired,amount1Desired) (contracts/yield-integrations/UniswapIntegration.sol#424-430)
		- (success,liquidityPositionId) = uniswapIntegrationDeployer.mintLiquidityPosition(liquidityPositions[liquidityPositionKeys[liquidityPositionIndex]].token0,liquidityPositions[liquidityPositionKeys[liquidityPositionIndex]].token1,liquidityPositions[liquidityPositionKeys[liquidityPositionIndex]].feeNumerator,liquidityPositions[liquidityPositionKeys[liquidityPositionIndex]].tickLower,liquidityP
ositions[liquidityPositionKeys[liquidityPositionIndex]].tickUpper,amount0Desired_scope_0,amount1Desired_scope_1) (contracts/yield-integrations/UniswapIntegration.sol#443-456)
	External calls sending eth:
	- transferTokensToUniswapIntegrationDeployer() (contracts/yield-integrations/UniswapIntegration.sol#294)
		- (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#139)
	State variables written after the call(s):
	- increaseLiquidityPositions(liquidityPositionDesiredDeployAmounts) (contracts/yield-integrations/UniswapIntegration.sol#346)
		- liquidityPositions[liquidityPositionKeys[liquidityPositionIndex]].minted = true (contracts/yield-integrations/UniswapIntegration.sol#457-458)
		- liquidityPositions[liquidityPositionKeys[liquidityPositionIndex]].id = liquidityPositionId (contracts/yield-integrations/UniswapIntegration.sol#459)
Reentrancy in DynamicRangeOrdersIntegration.rebalance() (contracts/yield-integrations/DynamicRangeOrdersIntegration.sol#358-362):
	External calls:
	- transferTokensToDynamicRangeOrdersIntegrationDeployer() (contracts/yield-integrations/DynamicRangeOrdersIntegration.sol#359)
		- returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/utils/SafeERC20Upgradeable.sol#71)
		- (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#139)
		- dynamicRangeOrdersIntegrationDeployer.tokenApprovals(address(token)) (contracts/yield-integrations/DynamicRangeOrdersIntegration.sol#382)
		- token.safeTransferFrom(address(dynamicRangeOrdersIntegrationDeployer),token.balanceOf(address(this))) (contracts/yield-integrations/DynamicRangeOrdersIntegration.sol#386-389)
	- closeExcessLiquidityPositions() (contracts/yield-integrations/DynamicRangeOrdersIntegration.sol#360)
		- dynamicRangeOrdersIntegrationDeployer.decreaseLiquidityPosition(liquidityPositionKeys[liquidityPositionIndex],positionActualBaseStablecoinValue - positionDesiredBaseStablecoinValue) (contracts/yield-integrations/DynamicRangeOrdersIntegration.sol#541-544)
	- _deploy() (contracts/yield-integrations/DynamicRangeOrdersIntegration.sol#361)
		- returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/utils/SafeERC20Upgradeable.sol#71)
		- (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#139)
		- dynamicRangeOrdersIntegrationDeployer.tokenApprovals(address(token)) (contracts/yield-integrations/DynamicRangeOrdersIntegration.sol#382)
		- dynamicRangeOrdersIntegrationDeployer.increaseLiquidityPosition(liquidityPositions[liquidityPositionKeys[liquidityPositionIndex]].id,liquidityPositions[liquidityPositionKeys[liquidityPositionIndex]].token0,liquidityPositions[liquidityPositionKeys[liquidityPositionIndex]].token1,amount0Desired,amount
1Desired) (contracts/yield-integrations/DynamicRangeOrdersIntegration.sol#415-424)
		- token.safeTransferFrom(address(dynamicRangeOrdersIntegrationDeployer),token.balanceOf(address(this))) (contracts/yield-integrations/DynamicRangeOrdersIntegration.sol#386-389)
		- (success,liquidityPositionId) = dynamicRangeOrdersIntegrationDeployer.mintLiquidityPosition(liquidityPositions[liquidityPositionKeys[liquidityPositionIndex]].token0,liquidityPositions[liquidityPositionKeys[liquidityPositionIndex]].token1,liquidityPositions[liquidityPositionKeys[liquidityPositionInde
x]].feeNumerator,liquidityPositions[liquidityPositionKeys[liquidityPositionIndex]].tickLower,liquidityPositions[liquidityPositionKeys[liquidityPositionIndex]].tickUpper,amount0Desired_scope_0,amount1Desired_scope_1) (contracts/yield-integrations/DynamicRangeOrdersIntegration.sol#437-455)
		- dynamicRangeOrdersIntegrationDeployer.swapExcessTokensForBaseStablecoin(tokenDesiredAmounts) (contracts/yield-integrations/DynamicRangeOrdersIntegration.sol#326-328)
		- dynamicRangeOrdersIntegrationDeployer.swapExcessBaseStablecoinForTokens(tokenDesiredAmounts) (contracts/yield-integrations/DynamicRangeOrdersIntegration.sol#331-333)
	External calls sending eth:
	- transferTokensToDynamicRangeOrdersIntegrationDeployer() (contracts/yield-integrations/DynamicRangeOrdersIntegration.sol#359)
		- (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#139)
	- _deploy() (contracts/yield-integrations/DynamicRangeOrdersIntegration.sol#361)
		- (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#139)
	State variables written after the call(s):
	- _deploy() (contracts/yield-integrations/DynamicRangeOrdersIntegration.sol#361)
		- liquidityPositions[liquidityPositionKeys[liquidityPositionIndex]].minted = true (contracts/yield-integrations/DynamicRangeOrdersIntegration.sol#457-458)
		- liquidityPositions[liquidityPositionKeys[liquidityPositionIndex]].id = liquidityPositionId (contracts/yield-integrations/DynamicRangeOrdersIntegration.sol#459-460)
```

```
Reentrancy in UniswapIntegration.rebalance() (contracts/yield-integrations/UniswapIntegration.sol#370-374):
        External calls:
        - transferTokensToUniswapIntegrationDeployer() (contracts/yield-integrations/UniswapIntegration.sol#371)
                - returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/utils/SafeERC20Upgradeable.sol#71)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#139)
                - uniswapIntegrationDeployer.tokenApprovals[address(token)] (contracts/yield-integrations/UniswapIntegration.sol#393)
                - token.safeTransferFrom(address(uniswapIntegrationDeployer),token,balanceOf(address(this))) (contracts/yield-integrations/UniswapIntegration.sol#396-399)
        - closeExcessLiquidityPositions() (contracts/yield-integrations/UniswapIntegration.sol#372)
                - uniswapIntegrationDeployer.decreaseLiquidityPosition(liquidityPositionIndex,positionActualBaseStablecoinValue - positionDesiredBaseStablecoinValue) (contracts/yield-integrations/UniswapIntegration.sol#487-498)
        - _deploy() (contracts/yield-integrations/UniswapIntegration.sol#373)
                - returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/utils/SafeERC20Upgradeable.sol#71)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#139)
                - uniswapIntegrationDeployer.tokenApprovals[address(token)] (contracts/yield-integrations/UniswapIntegration.sol#393)
                - uniswapIntegrationDeployer.increaseLiquidityPosition(liquidityPositions[liquidityPositionIndex].id,liquidityPositions[liquidityPositionIndex].token0,liquidityPositions[liquidityPositionIndex].token1,amount0Desired,amount1Desired) (contracts/yield-integrations/UniswapIntegration.sol#424-438)
                - token.safeTransfer(address(uniswapIntegrationDeployer),token.balanceOf(address(this))) (contracts/yield-integrations/UniswapIntegration.sol#396-399)
                - (success,liquidityPositionId) = uniswapIntegrationDeployer.mintLiquidityPosition(liquidityPositions[liquidityPositionIndex].token0,liquidityPositions[liquidityPositionIndex].token1,liquidityPositions[liquidityPositionIndex].feeNumerator,liquidityPositions[liquidityPositionIndex].tickLower,liquidityPositions[liquidityPositionIndex].tickUpper,amount0Desired_scope_0,amount1Desired_scope_1) (contracts/yield-integrations/UniswapIntegration.sol#443-454)
                - uniswapIntegrationDeployer.swapExcessTokensForBaseStablecoin(tokenDesiredAmounts) (contracts/yield-integrations/UniswapIntegration.sol#338-340)
                - uniswapIntegrationDeployer.swapExcessBaseStablecoinForTokens(tokenDesiredAmounts) (contracts/yield-integrations/UniswapIntegration.sol#343-345)
        External calls sending eth:
        - transferTokensToUniswapIntegrationDeployer() (contracts/yield-integrations/UniswapIntegration.sol#371)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#119)
        - _deploy() (contracts/yield-integrations/UniswapIntegration.sol#373)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#119)
        State variables written after the call(s):
        - _deploy() (contracts/yield-integrations/UniswapIntegration.sol#373)
                - liquidityPositions[liquidityPositionIndex].minted = true (contracts/yield-integrations/UniswapIntegration.sol#458)
                - liquidityPositions[liquidityPositionIndex].id = liquidityPositionId (contracts/yield-integrations/UniswapIntegration.sol#459)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities

AccessControlEnumerableUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/access/AccessControlEnumerableUpgradeable.sol#97) shadows:
        - AccessControlUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/access/AccessControlUpgradeable.sol#252)
        - ERC165Upgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/utils/introspection/ERC165Upgradeable.sol#43)
        - ContextUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#21)
AccessControlUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/access/AccessControlUpgradeable.sol#252) shadows:
        - ERC165Upgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/utils/introspection/ERC165Upgradeable.sol#35)
        - ContextUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#91)
OwnableUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol#76) shadows:
        - ContextUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#91)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variable-shadowing

YieldManager.ethToProtocolFeeAccrual(uint256) (contracts/core/YieldManager.sol#656-688) ignores return value by IERC20MetadataUpgradeable(wethAddress).transfer(moduleMap.getModuleAddress(Modules.Kernel),protocolFeeEthRewardsAmount) (contracts/core/YieldManager.sol#682-685)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-transfer

MasterChef.safeSushiTransfer(address,uint256) (contracts/mock/MasterChef.sol#1571-1578) ignores return value by sushi.transfer(_to,sushiBal) (contracts/mock/MasterChef.sol#1576)
MasterChef.safeSushiTransfer(address,uint256) (contracts/mock/MasterChef.sol#1571-1578) ignores return value by sushi.transfer(_to,_amount) (contracts/mock/MasterChef.sol#1576)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-transfer

DynamicRangeOrdersIntegration.liquidityPositionKeyIndexes (contracts/yield-integrations/DynamicRangeOrdersIntegration.sol#67) is never initialized. It is used in:
        - DynamicRangeOrdersIntegration.rerangeLiquidityPosition(bytes32) (contracts/yield-integrations/DynamicRangeOrdersIntegration.sol#476-518)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-state-variables
```

```
FullMath.mulDiv(uint256,uint256,uint256) (contracts/libraries/FullMath.sol#14-106) performs a multiplication on the result of a division:
        -denominator = denominator / twos (contracts/libraries/FullMath.sol#67)
        -inv = (3 * denominator) ^ 2 (contracts/libraries/FullMath.sol#87)
FullMath.mulDiv(uint256,uint256,uint256) (contracts/libraries/FullMath.sol#14-106) performs a multiplication on the result of a division:
        -denominator = denominator / twos (contracts/libraries/FullMath.sol#67)
        -inv *= 2 - denominator * inv (contracts/libraries/FullMath.sol#91)
FullMath.mulDiv(uint256,uint256,uint256) (contracts/libraries/FullMath.sol#14-106) performs a multiplication on the result of a division:
        -denominator = denominator / twos (contracts/libraries/FullMath.sol#67)
        -inv *= 2 - denominator * inv (contracts/libraries/FullMath.sol#92)
FullMath.mulDiv(uint256,uint256,uint256) (contracts/libraries/FullMath.sol#14-106) performs a multiplication on the result of a division:
        -denominator = denominator / twos (contracts/libraries/FullMath.sol#67)
        -inv *= 2 - denominator * inv (contracts/libraries/FullMath.sol#93)
FullMath.mulDiv(uint256,uint256,uint256) (contracts/libraries/FullMath.sol#14-106) performs a multiplication on the result of a division:
        -denominator = denominator / twos (contracts/libraries/FullMath.sol#67)
        -inv *= 2 - denominator * inv (contracts/libraries/FullMath.sol#94)
FullMath.mulDiv(uint256,uint256,uint256) (contracts/libraries/FullMath.sol#14-106) performs a multiplication on the result of a division:
        -denominator = denominator / twos (contracts/libraries/FullMath.sol#67)
        -inv *= 2 - denominator * inv (contracts/libraries/FullMath.sol#95)
FullMath.mulDiv(uint256,uint256,uint256) (contracts/libraries/FullMath.sol#14-106) performs a multiplication on the result of a division:
        -denominator = denominator / twos (contracts/libraries/FullMath.sol#67)
        -inv *= 2 - denominator * inv (contracts/libraries/FullMath.sol#96)
FullMath.mulDiv(uint256,uint256,uint256) (contracts/libraries/FullMath.sol#14-106) performs a multiplication on the result of a division:
        -prod0 = prod0 / twos (contracts/libraries/FullMath.sol#72)
        -result = prod0 * inv (contracts/libraries/FullMath.sol#104)
SushiSwapIntegration.getPoolEthValue(uint256) (contracts/yield-integrations/SushiSwapIntegration.sol#124-148) performs a multiplication on the result of a division:
        -tokenAmount = sharePercent / 2 * IERC20(token0).balanceOf(address(pools[poolId].poolInfo.lpToken)) / 1000000000 (contracts/yield-integrations/SushiSwapIntegration.sol#134)
SushiSwapIntegration.getPoolEthValue(uint256) (contracts/yield-integrations/SushiSwapIntegration.sol#124-148) performs a multiplication on the result of a division:
        -tokenAmount = sharePercent / 2 * IERC20(token1).balanceOf(address(pools[poolId].poolInfo.lpToken)) / 1000000000 (contracts/yield-integrations/SushiSwapIntegration.sol#135)
SushiSwapIntegration.getPriceWithMultiplier(address,address,address) (contracts/yield-integrations/SushiSwapIntegration.sol#242-257) performs a multiplication on the result of a division:
        -priceWithMultiplier = (MULTIPLIER * (reserve1 / 10 ** IERC20(address(wethAddress)).decimals())) / (reserve0 / 10 ** IERC20(address(tokenToAdd)).decimals()) (contracts/yield-integrations/SushiSwapIntegration.sol#249-252)
SushiSwapIntegration.getPriceWithMultiplier(address,address,address) (contracts/yield-integrations/SushiSwapIntegration.sol#242-257) performs a multiplication on the result of a division:
        -priceWithMultiplier = (MULTIPLIER * (reserve0 / 10 ** IERC20(address(tokenToAdd)).decimals())) / (reserve1 / 10 ** IERC20(address(wethAddress)).decimals()) (contracts/yield-integrations/SushiSwapIntegration.sol#249-252)
SushiSwapIntegration.calculateAmountOfTokenToAddLiquidityETH(uint256,uint256) (contracts/yield-integrations/SushiSwapIntegration.sol#259-269) performs a multiplication on the result of a division:
        -amountTokenDesired = amountWeis * priceWithMultiplier / MULTIPLIER (contracts/yield-integrations/SushiSwapIntegration.sol#264)
        -amountTokenMin = amountTokenDesired - CamountTokenDesired * 1) / 100 (contracts/yield-integrations/SushiSwapIntegration.sol#265)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply

MasterChef.pendingSushi(uint256,address) (contracts/mock/MasterChef.sol#1491-1502) performs a multiplication on the result of a division:
        -sushiReward = multiplier.mul(sushiPerBlock).mul(pool.allocPoint).div(totalAllocPoint) (contracts/mock/MasterChef.sol#1498)
        -accSushiPerShare = accSushiPerShare.add(sushiReward.mul(1e12).div(lpSupply)) (contracts/mock/MasterChef.sol#1499)
MasterChef.updatePool(uint256) (contracts/mock/MasterChef.sol#1513-1529) performs a multiplication on the result of a division:
        -sushiReward = multiplier.mul(sushiPerBlock).mul(pool.allocPoint).div(totalAllocPoint) (contracts/mock/MasterChef.sol#1524)
        -pool.accSushiPerShare = pool.accSushiPerShare.add(sushiReward.mul(1e12).div(lpSupply)) (contracts/mock/MasterChef.sol#1527)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply

SushiSwapIntegration._deploy(uint256,uint256) (contracts/yield-integrations/SushiSwapIntegration.sol#171-233) uses a dangerous strict equality:
        - tokensReceived == 0 (contracts/yield-integrations/SushiSwapIntegration.sol#203)
SushiSwapIntegration._deploy(uint256,uint256) (contracts/yield-integrations/SushiSwapIntegration.sol#171-233) uses a dangerous strict equality:
        - tokensReceived == 0 (contracts/yield-integrations/SushiSwapIntegration.sol#205)
SushiSwapIntegration.getPoolEthValue(uint256) (contracts/yield-integrations/SushiSwapIntegration.sol#124-148) uses a dangerous strict equality:
        - sharePercent == 0 (contracts/yield-integrations/SushiSwapIntegration.sol#130)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities
```

```
MasterChef.pendingSushi(uint256,address) (contracts/mock/MasterChef.sol#1491-1502) performs a multiplication on the result of a division:
        -sushiReward = multiplier.mul(sushiPerBlock).mul(pool.allocPoint).div(totalAllocPoint) (contracts/mock/MasterChef.sol#1498)
        -accSushiPerShare = accSushiPerShare.add(sushiReward.mul(1e12).div(lpSupply)) (contracts/mock/MasterChef.sol#1499)
MasterChef.updatePool(uint256) (contracts/mock/MasterChef.sol#1513-1529) performs a multiplication on the result of a division:
        -sushiReward = multiplier.mul(sushiPerBlock).mul(pool.allocPoint).div(totalAllocPoint) (contracts/mock/MasterChef.sol#1524)
        -pool.accSushiPerShare = pool.accSushiPerShare.add(sushiReward.mul(1e12).div(lpSupply)) (contracts/mock/MasterChef.sol#1527)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply

SushiSwapIntegration._deploy(uint256,uint256) (contracts/yield-integrations/SushiSwapIntegration.sol#171-233) uses a dangerous strict equality:
        - tokensReceived == 0 (contracts/yield-integrations/SushiSwapIntegration.sol#203)
SushiSwapIntegration._deploy(uint256,uint256) (contracts/yield-integrations/SushiSwapIntegration.sol#171-233) uses a dangerous strict equality:
        - tokensReceived == 0 (contracts/yield-integrations/SushiSwapIntegration.sol#205)
SushiSwapIntegration.getPoolEthValue(uint256) (contracts/yield-integrations/SushiSwapIntegration.sol#124-148) uses a dangerous strict equality:
        - sharePercent == 0 (contracts/yield-integrations/SushiSwapIntegration.sol#130)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities

SushiToken._writeCheckpoint(address,uint32,uint256,uint256) (contracts/mock/MasterChef.sol#1307-1325) uses a dangerous strict equality:
        - nCheckpoints > 0 && checkpoints[delegatee][nCheckpoints - 1].fromBlock == blockNumber (contracts/mock/MasterChef.sol#1317)
MasterChef.add(uint256) (contracts/mock/MasterChef.sol#1448-1475) uses a dangerous strict equality:
        - require(bool,string)(bal == newUpToken.balanceOf(address(this)),migrate: bad) (contracts/mock/MasterChef.sol#1473)
MasterChef.updatePool(uint256) (contracts/mock/MasterChef.sol#1513-1529) uses a dangerous strict equality:
        - lpSupply == 0 (contracts/mock/MasterChef.sol#1519)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities

StrategyMap.deleteStrategy(uint256) (contracts/core/StrategyMap.sol#195-226) deletes IStrategyMap.Strategy (contracts/interfaces/IStrategyMap.sol#12-18) which contains a mapping:
        -delete strategies[id] (contracts/core/StrategyMap.sol#218)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#deletion-on-mapping-containing-a-structure

Reentrancy in UserPositions._withdraw(address,address[],uint256[],bool) (contracts/core/UserPositions.sol#233-304):
        External calls:
        - closePositionsForWithdrawal(tokens[token0],amounts[token0]) (contracts/core/UserPositions.sol#249)
                - integration.withdraw(token,integration.getBalance(token)) (contracts/core/UserPositions.sol#370)
                - IIntegration(integrationAddress).withdraw(token,desiredWithdrawAmount) (contracts/core/UserPositions.sol#348)
        - IERC20MetadataUpgradeable(tokens[tokenId]).safeTransferFrom(moduleMap.getModuleAddress(Modules.Kernel),recipient,amounts[tokenId]) (contracts/core/UserPositions.sol#286-290)
        - IRinkRewards(moduleMap.getModuleAddress(Modules.RinkRewards)).decreaseRewards(tokens[tokenId],recipient,amounts[tokenId]) (contracts/core/UserPositions.sol#296-295)
        - IEtherRewards(moduleMap.getModuleAddress(Modules.EtherRewards)).updateUserRewards(tokens[tokenId],recipient) (contracts/core/UserPositions.sol#297-298)
        State variables written after the call(s):
        - _balances[tokens[tokenId]][recipient] -= amounts[tokenId] (contracts/core/UserPositions.sol#302)
Reentrancy in YieldManager.ethToRewards(uint256) (contracts/core/YieldManager.sol#416-452):
        External calls:
        - IEtherRewards(moduleMap.getModuleAddress(Modules.EtherRewards)).increaseEthRewards(tokenAddress,ethRewardsAmount * processedWethByToken[tokenAddress]) / processedWethBySum) (contracts/core/YieldManager.sol#433-438)
        State variables written after the call(s):
        - processedWethByToken[tokenAddress] = 0 (contracts/core/YieldManager.sol#448)
Reentrancy in DynamicRangeOrdersIntegration.increaseLiquidityPositions(uint256[]) (contracts/yield-integrations/DynamicRangeOrdersIntegration.sol#395-465):
        External calls:
        - (success,liquidityPositionId) = dynamicRangeOrdersIntegrationDeployer.mintLiquidityPosition(liquidityPositions[liquidityPositionIndex]).token0,liquidityPositions[liquidityPositionIndex].token1,liquidityPositions[liquidityPositionKeys[liquidityPositionIndex]].feeNumerator,liquidityPositions[liquidityPositionKeys[liquidityPositionIndex]].tickLower,liquidityPositions[liquidityPositionKeys[liquidityPositionIndex]].tickUpper,amount0Desired_scope_0,amount1Desired_scope_1) (contracts/yield-integrations/DynamicRangeOrdersIntegration.sol#437-455)
        State variables written after the call(s):
        - liquidityPositions[liquidityPositionKeys[liquidityPositionIndex]].minted = true (contracts/yield-integrations/DynamicRangeOrdersIntegration.sol#457-458)
        - liquidityPositions[liquidityPositionKeys[liquidityPositionIndex]].id = liquidityPositionId (contracts/yield-integrations/DynamicRangeOrdersIntegration.sol#459-460)
Reentrancy in UniswapIntegration.increaseLiquidityPositions(uint256[]) (contracts/yield-integrations/UniswapIntegration.sol#405-464):
        External calls:
        - (success,liquidityPositionId) = uniswapIntegrationDeployer.mintLiquidityPosition(liquidityPositions[liquidityPositionIndex].token0,liquidityPositions[liquidityPositionIndex].token1,liquidityPositions[liquidityPositionIndex].feeNumerator,liquidityPositions[liquidityPositionIndex].tickLower,liquidityPositions[liquidityPositionIndex].tickUpper,amount0Desired_scope_0,amount1Desired_scope_1) (contracts/yield-integrations/UniswapIntegration.sol#443-454)
        State variables written after the call(s):
        - liquidityPositions[liquidityPositionIndex].minted = true (contracts/yield-integrations/UniswapIntegration.sol#458)
        - liquidityPositions[liquidityPositionIndex].id = liquidityPositionId (contracts/yield-integrations/UniswapIntegration.sol#459)
```

```
Reentrancy in DynamicRangeOrdersIntegration.rerangeLiquidityPosition(bytes32) (contracts/yield-integrations/DynamicRangeOrdersIntegration.sol#476-518):
        External calls:
        - dynamicRangeOrdersIntegrationDeployer.decreaseLiquidityPosition(liquidityPositionKey,positionActualBaseStableceivnValue) (contracts/yield-integrations/DynamicRangeOrdersIntegration.sol#483-486)
        - (success,liquidityPositionId) = dynamicRangeOrdersIntegrationDeployer.mintLiquidityPosition(liquidityPositions[liquidityPositionKey].token0,liquidityPositions[liquidityPositionKey].token1,liquidityPositions[liquidityPositionKey].fee.toUint24(),liquidityPositions[liquidityPositionKey].tickLower,liquidityPositions[liquidityPositionKey].tickUpper,amount0Desired,amount1Desired) (contracts/yield-integrations/DynamicRangeOrdersIntegration.sol#497-508)
        State variables written after the call(s):
        - liquidityPositions[liquidityPositionKey].minted = true (contracts/yield-integrations/DynamicRangeOrdersIntegration.sol#512)
        - liquidityPositions[liquidityPositionKey].id = liquidityPositionId (contracts/yield-integrations/DynamicRangeOrdersIntegration.sol#513)
Reentrancy in UserPositions.transferToStrategy(address,address[],uint256[]) (contracts/core/UserPositions.sol#394-428):
        External calls:
        - IBioRewards(moduleMap.getModuleAddress(Modules.BioRewards)).decreaseRewards(tokens[tokenId],recipient,amounts[tokenId]) (contracts/core/UserPositions.sol#418-419)
        - IEtherRewards(moduleMap.getModuleAddress(Modules.EtherRewards)).updateUserRewards(tokens[tokenId],recipient) (contracts/core/UserPositions.sol#421-422)
        State variables written after the call(s):
        - _balances[tokens[tokenId][recipient] -= amounts[tokenId] (contracts/core/UserPositions.sol#426)
Reentrancy in AaveIntegration.withdraw(address,uint256) (contracts/yield-integrations/AaveIntegration.sol#52-85):
        External calls:
        - IAaveLendingPool(lendingPoolAddress).withdraw(tokenAddress,amount,address(this)) (contracts/yield-integrations/AaveIntegration.sol#65-71)
        State variables written after the call(s):
        - balances[tokenAddress] -= amount (contracts/yield-integrations/AaveIntegration.sol#80)
Reentrancy in YearnIntegration.withdraw(address,uint256) (contracts/yield-integrations/YearnIntegration.sol#52-98):
        External calls:
        - IYearnVault(vaultAddress).withdraw(sharesAmount) (contracts/yield-integrations/YearnIntegration.sol#86)
        State variables written after the call(s):
        - balances[tokenAddress] -= amount (contracts/yield-integrations/YearnIntegration.sol#96)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1

Reentrancy in MasterChef.add(uint256,IERC20,bool) (contracts/mock/MasterChef.sol#1437-1449):
        External calls:
        - massUpdatePools() (contracts/mock/MasterChef.sol#1439)
                - sushi.mint(devaddr,sushiReward.div(10)) (contracts/mock/MasterChef.sol#1525)
                - sushi.mint(address(this),sushiReward) (contracts/mock/MasterChef.sol#1526)
        State variables written after the call(s):
        - poolInfo.push(PoolInfo({lpToken:_lpToken,_allocPoint,lastRewardBlock,0}) (contracts/mock/MasterChef.sol#1443-1448)
        - totalAllocPoint = totalAllocPoint.add(_allocPoint) (contracts/mock/MasterChef.sol#1442)
Reentrancy in MasterChef.deposit(uint256,uint256) (contracts/mock/MasterChef.sol#1532-1544):
        External calls:
        - updatePool(_pid) (contracts/mock/MasterChef.sol#1535)
                - sushi.mint(devaddr,sushiReward.div(10)) (contracts/mock/MasterChef.sol#1525)
                - sushi.mint(address(this),sushiReward) (contracts/mock/MasterChef.sol#1526)
        - safeSushiTransfer(msg.sender,pending) (contracts/mock/MasterChef.sol#1538)
                - sushi.transfer(_to,sushiBal) (contracts/mock/MasterChef.sol#1574)
                - sushi.transfer(_to,_amount) (contracts/mock/MasterChef.sol#1576)
        - pool.lpToken.safeTransferFrom(address(msg.sender),address(this),_amount) (contracts/mock/MasterChef.sol#1540)
        State variables written after the call(s):
        - user.amount = user.amount.add(_amount) (contracts/mock/MasterChef.sol#1541)
        - user.rewardDebt = user.amount.mul(pool.accSushiPerShare).div(1e12) (contracts/mock/MasterChef.sol#1542)
Reentrancy in MasterChef.emergencyWithdraw(uint256) (contracts/mock/MasterChef.sol#1561-1568):
        External calls:
        - pool.lpToken.safeTransfer(address(msg.sender),user.amount) (contracts/mock/MasterChef.sol#1564)
        State variables written after the call(s):
        - user.amount = 0 (contracts/mock/MasterChef.sol#1566)
        - user.rewardDebt = 0 (contracts/mock/MasterChef.sol#1567)
Reentrancy in MasterChef.migrate(uint256) (contracts/mock/MasterChef.sol#1446-1475):
        External calls:
        - lpToken.safeApprove(address(migrator),bal) (contracts/mock/MasterChef.sol#1471)
        - newLpToken = migrator.migrate(lpToken) (contracts/mock/MasterChef.sol#1472)
        State variables written after the call(s):
        - pool.lpToken = newLpToken (contracts/mock/MasterChef.sol#1474)
```

```
YieldManager.processYield().tokenId (contracts/core/YieldManager.sol#277) is a local variable never initialized
SushiSwapIntegration.getAllPositionsETHValue().i (contracts/yield-integrations/SushiSwapIntegration.sol#146) is a local variable never initialized
DynamicRangeOrdersIntegrationDeployer.swapExcessBaseStableceivnForTokens(uint256[]).tokenId (contracts/yield-integrations/DynamicRangeOrdersIntegrationDeployer.sol#135) is a local variable never initialized
DynamicRangeOrdersIntegration.closeExcessLiquidityPositions().liquidityPositionIndex (contracts/yield-integrations/DynamicRangeOrdersIntegration.sol#527) is a local variable never initialized
SushiSwapIntegration._deploy(uint256,uint256).tokensReceived (contracts/yield-integrations/SushiSwapIntegration.sol#177) is a local variable never initialized
UniswapTrader._swapExactOutput(address,address,address,uint256).exactOutputSingleParams (contracts/core/UniswapTrader.sol#417) is a local variable never initialized
SushiSwapIntegration.withdraw(address,uint256).result (contracts/yield-integrations/SushiSwapIntegration.sol#113) is a local variable never initialized
UniswapIntegration._deploy().liquidityPositionIndex (contracts/yield-integrations/UniswapIntegration.sol#201) is a local variable never initialized
UniswapIntegration.getAllPositionsBaseStableceivnValue().liquidityPositionIndex (contracts/yield-integrations/UniswapIntegration.sol#488) is a local variable never initialized
DynamicRangeOrdersIntegrationDeployer.harvestYield(bytes32[]).totalBalanceBaseStableceivnValue (contracts/yield-integrations/DynamicRangeOrdersIntegrationDeployer.sol#538) is a local variable never initialized
UniswapIntegrationDeployer.collectTokensFromPosition(uint256,uint128,uint128).returnedAmount0 (contracts/yield-integrations/UniswapIntegrationDeployer.sol#643) is a local variable never initialized
UniswapTrader.addPool(address,address,uint24,uint24).newTokenPair (contracts/core/UniswapTrader.sol#148) is a local variable never initialized
UserPositions.fullyCloseAllPositionsFromAddress(address,uint256).doneClosingPositions (contracts/core/UserPositions.sol#367) is a local variable never initialized
UniswapIntegration.transferTokensToUniswapIntegrationDeployer().tokenId (contracts/yield-integrations/UniswapIntegration.sol#303) is a local variable never initialized
DynamicRangeOrdersIntegration.transferTokensToDynamicRangeOrdersIntegrationDeployer().tokenId (contracts/yield-integrations/DynamicRangeOrdersIntegration.sol#371) is a local variable never initialized
UniswapTrader.addPool(address,address,uint24,uint24).tokenPairAdded (contracts/core/UniswapTrader.sol#137) is a local variable never initialized
UserPositions._withdraw(address,address[],uint256[],bool).tokenId (contracts/core/UserPositions.sol#249) is a local variable never initialized
DynamicRangeOrdersIntegrationDeployer.mintLiquidityPosition(address,address,uint24,int24,int24,uint256,uint256).mintParams (contracts/yield-integrations/DynamicRangeOrdersIntegrationDeployer.sol#216) is a local variable never initialized
Controlled.addController(address).added (contracts/core/Controlled.sol#25) is a local variable never initialized
UniswapTrader.addPool(address,address,uint24).pairIndex (contracts/core/UniswapTrader.sol#136) is a local variable never initialized
BioRewards.getUserBioRewards(address).tokenId (contracts/core/BioRewards.sol#151-152) is a local variable never initialized
UniswapIntegration.increaseLiquidityPositions(uint256[]).amountDesired_scope_1 (contracts/yield-integrations/UniswapIntegration.sol#424) is a local variable never initialized
UniswapIntegration.configureLiquidityPosition(address,address,uint24,int24,int24,uint256).liquidityPosition (contracts/yield-integrations/UniswapIntegration.sol#147) is a local variable never initialized
UniswapIntegrationDeployer.mintLiquidityPosition(address,address,uint24,int24,int24,uint256,uint256).mintParams (contracts/yield-integrations/UniswapIntegrationDeployer.sol#214) is a local variable never initialized
YieldManager.getProcessedOwnShw(tokenShw).tokenId (contracts/core/YieldManager.sol#633) is a local variable never initialized
UserPositions.transferToStrategy(address,address[],uint256[]).tokenId (contracts/core/UserPositions.sol#405) is a local variable never initialized
UniswapIntegrationDeployer.harvestYield().liquidityPositionIndex (contracts/yield-integrations/UniswapIntegrationDeployer.sol#587) is a local variable never initialized
YieldManager.harvestYield().tokenId (contracts/core/YieldManager.sol#222) is a local variable never initialized
DynamicRangeOrdersIntegrationDeployer.increaseLiquidityPosition(uint256,address,address,uint256,uint256).increaseLiquidityParams (contracts/yield-integrations/DynamicRangeOrdersIntegrationDeployer.sol#271-272) is a local variable never initialized
UniswapIntegration.getBaseStableceivnReserveBalance().tokenId (contracts/yield-integrations/UniswapIntegration.sol#555) is a local variable never initialized
UniswapIntegrationDeployer.mintLiquidityPosition(address,address,uint24,int24,int24,uint256,uint256).returnedLiquidityPositionId (contracts/yield-integrations/UniswapIntegrationDeployer.sol#238) is a local variable never initialized
UniswapIntegrationDeployer.closeAllPositionsWithdrawal(address,uint256).doneClosingPositions (contracts/yield-integrations/UniswapIntegrationDeployer.sol#308) is a local variable never initialized
DynamicRangeOrdersIntegrationDeployer.mintLiquidityPosition(address,address,uint24,int24,int24,uint256,uint256).returnedLiquidityPositionId (contracts/yield-integrations/DynamicRangeOrdersIntegrationDeployer.sol#232) is a local variable never initialized
DynamicRangeOrdersIntegrationDeployer.harvestYield(bytes32[]).liquidityPositionIndex (contracts/yield-integrations/DynamicRangeOrdersIntegrationDeployer.sol#585) is a local variable never initialized
DynamicRangeOrdersIntegrationDeployer.collectTokensFromPosition(uint256,uint128,uint128).returnedAmount1 (contracts/yield-integrations/DynamicRangeOrdersIntegrationDeployer.sol#643) is a local variable never initialized
UniswapTrader.addPool(address,address,uint24,uint24).poolIndex (contracts/core/UniswapTrader.sol#121) is a local variable never initialized
YearnIntegration.getValueInEth(address).vaultAddress (contracts/yield-integrations/YearnIntegration.sol#143) is a local variable never initialized
UserPositions._claimBioRewards(address).tokenId (contracts/core/UserPositions.sol#574) is a local variable never initialized
EtherRewards.getUserEtherRewards(address).tokenId (contracts/core/EtherRewards.sol#141) is a local variable never initialized
SushiSwapIntegration.withdraw(address,uint256).poolIndex (contracts/yield-integrations/SushiSwapIntegration.sol#115) is a local variable never initialized
DynamicRangeOrdersIntegration.increaseLiquidityPositions(uint256[]).amount0Desired_scope_0 (contracts/yield-integrations/DynamicRangeOrdersIntegration.sol#428) is a local variable never initialized
StrategyMap.updateTokens(uint256,address[]).i (contracts/core/StrategyMap.sol#110) is a local variable never initialized
UniswapIntegrationDeployer.increaseLiquidityPosition(uint256,address,address,uint256,uint256).increaseLiquidityParams (contracts/yield-integrations/UniswapIntegrationDeployer.sol#260-270) is a local variable never initialized
UniswapIntegrationDeployer.harvestYield().tokenId_scope_0 (contracts/yield-integrations/UniswapIntegrationDeployer.sol#589) is a local variable never initialized
StrategyMap._getStrategySummary(uint256).result (contracts/core/StrategyMap.sol#267) is a local variable never initialized
DynamicRangeOrdersIntegrationDeployer.harvestYield(bytes32[]).tokenId (contracts/yield-integrations/DynamicRangeOrdersIntegrationDeployer.sol#564) is a local variable never initialized
UniswapTrader.swapExactInput(address,address,address,uint256).exactInputSingleParams (contracts/core/UniswapTrader.sol#338) is a local variable never initialized
Controlled._Controlled_init(address[],address).i (contracts/core/Controlled.sol#173) is a local variable never initialized
UniswapIntegrationDeployer.harvestYield().totalBalanceBaseStableceivnValue (contracts/yield-integrations/UniswapIntegrationDeployer.sol#538) is a local variable never initialized
UniswapIntegrationDeployer.collectTokensFromPosition(uint256,uint128,uint128).collectParams (contracts/yield-integrations/UniswapIntegrationDeployer.sol#634) is a local variable never initialized
DynamicRangeOrdersIntegration.increaseLiquidityPositions(uint256[]).liquidityPositionIndex (contracts/yield-integrations/DynamicRangeOrdersIntegration.sol#399) is a local variable never initialized
YieldManager.getTokenTotalIntegrationBalance(address).integrationId (contracts/core/YieldManager.sol#634) is a local variable never initialized
UniswapTrader.addPool(address,address,uint24,uint24).poolAdded (contracts/core/UniswapTrader.sol#139) is a local variable never initialized
UserPositions.deposit(address,address[],uint256[],uint256).tokenId (contracts/core/UserPositions.sol#188) is a local variable never initialized
DynamicRangeOrdersIntegrationDeployer.swapExcessTokensForBaseStableceivn(uint256[]).tokenId (contracts/yield-integrations/DynamicRangeOrdersIntegrationDeployer.sol#497) is a local variable never initialized
Controlled.addController(address).i (contracts/core/Controlled.sol#26) is a local variable never initialized
DynamicRangeOrdersIntegration._deploy().liquidityPositionIndex (contracts/yield-integrations/DynamicRangeOrdersIntegration.sol#208) is a local variable never initialized
DynamicRangeOrdersIntegrationDeployer.collectTokensFromPosition(uint256,uint128,uint128).returnedAmount0 (contracts/yield-integrations/DynamicRangeOrdersIntegrationDeployer.sol#643) is a local variable never initialized
```

```
UniswapTrader.addPool(address,address,uint24,uint24).newPool (contracts/core/UniswapTrader.sol#132) is a local variable never initialized
DynamicRangeOrdersIntegration.getBaseStableceivnReserveBalance().tokenId (contracts/yield-integrations/DynamicRangeOrdersIntegration.sol#609) is a local variable never initialized
EtherRewards.claimEthRewards(address).tokenId (contracts/core/EtherRewards.sol#86) is a local variable never initialized
UserPositions.closePositionsForWithdrawal(address,uint256).integrationId (contracts/core/UserPositions.sol#328) is a local variable never initialized
DynamicRangeOrdersIntegrationDeployer.decreaseLiquidityPosition(bytes32,uint256).decreaseLiquidityParams (contracts/yield-integrations/DynamicRangeOrdersIntegrationDeployer.sol#448-461) is a local variable never initialized
UserPositions.transferFromStrategy(address,address[],uint256[]).tokenId (contracts/core/UserPositions.sol#444) is a local variable never initialized
YieldManager.withToRewards(uint256).tokenId (contracts/core/YieldManager.sol#629) is a local variable never initialized
UniswapIntegration.increaseLiquidityPositions(uint256[]).amount0Desired_scope_0 (contracts/yield-integrations/UniswapIntegration.sol#435) is a local variable never initialized
UniswapIntegration.increaseLiquidityPositions(uint256[]).liquidityPositionIndex (contracts/yield-integrations/UniswapIntegration.sol#409) is a local variable never initialized
UniswapIntegrationDeployer.getLiquidityPositionWeightSum().liquidityPositionIndex (contracts/yield-integrations/UniswapIntegrationDeployer.sol#585) is a local variable never initialized
UserPositions.withdrawAllAndClaim(address,address[],bool).tokenId (contracts/core/UserPositions.sol#287) is a local variable never initialized
DynamicRangeOrdersIntegration.getLiquidityPositionWeightSum().liquidityPositionIndex (contracts/yield-integrations/DynamicRangeOrdersIntegration.sol#639) is a local variable never initialized
UniswapIntegrationDeployer.swapExcessTokensForBaseStableceivn(uint256[]).tokenId (contracts/yield-integrations/UniswapIntegrationDeployer.sol#495) is a local variable never initialized
StrategyMap.updateTokens(uint256,address[]).i_scope_0 (contracts/core/StrategyMap.sol#113) is a local variable never initialized
UniswapIntegrationDeployer.swapExcessBaseStableceivnForTokens(uint256[]).tokenId (contracts/yield-integrations/UniswapIntegrationDeployer.sol#133) is a local variable never initialized
UniswapIntegrationDeployer.decreaseLiquidityPosition(uint256,uint256).decreaseLiquidityParams (contracts/yield-integrations/UniswapIntegrationDeployer.sol#443-444) is a local variable never initialized
DynamicRangeOrdersIntegration.getAllPositionsBaseStableceivnValue().liquidityPositionIndex (contracts/yield-integrations/DynamicRangeOrdersIntegration.sol#667) is a local variable never initialized
YieldManager.getProcessedOwnShw().tokenId (contracts/core/YieldManager.sol#582) is a local variable never initialized
UniswapIntegrationDeployer.harvestYield().tokenId (contracts/yield-integrations/UniswapIntegrationDeployer.sol#563) is a local variable never initialized
UserPositions._increaseBioRewards().tokenId (contracts/core/UserPositions.sol#497) is a local variable never initialized
UniswapIntegrationDeployer.collectTokensFromPosition(uint256,uint128,uint128).returnedAmount1 (contracts/yield-integrations/UniswapIntegrationDeployer.sol#643) is a local variable never initialized
IntegrationMap.getBioRewardPoolShw().tokenId (contracts/core/IntegrationMap.sol#315) is a local variable never initialized
DynamicRangeOrdersIntegration.increaseLiquidityPositions(uint256[]).amount1Desired_scope_1 (contracts/yield-integrations/DynamicRangeOrdersIntegration.sol#430) is a local variable never initialized
SushiSwapIntegration.getPoolWeightSumC.poolIndex (contracts/yield-integrations/SushiSwapIntegration.sol#117) is a local variable never initialized
UniswapIntegration.closeExcessLiquidityPositions().liquidityPositionIndex (contracts/yield-integrations/UniswapIntegration.sol#479) is a local variable never initialized
DynamicRangeOrdersIntegration.configureLiquidityPosition(address,address,uint24,int24,int24,uint256).liquidityPosition (contracts/yield-integrations/DynamicRangeOrdersIntegration.sol#131) is a local variable never initialized
YieldManager.harvestYield().integrationId (contracts/core/YieldManager.sol#224) is a local variable never initialized
StrategyMap.updateIntegrations(uint256,IStrategyMap.WeightedIntegration[]).newStrategyTotalWeight (contracts/core/StrategyMap.sol#149) is a local variable never initialized
DynamicRangeOrdersIntegrationDeployer.collectTokensFromPosition(uint256,uint128,uint128).collectParams (contracts/yield-integrations/DynamicRangeOrdersIntegrationDeployer.sol#634) is a local variable never initialized
DynamicRangeOrdersIntegrationDeployer.harvestYield(bytes32[]).tokenId_scope_0 (contracts/yield-integrations/DynamicRangeOrdersIntegrationDeployer.sol#589) is a local variable never initialized
DynamicRangeOrdersIntegrationDeployer.closePositionsWithdrawal(bytes32[],address,uint256).doneClosingPositions (contracts/yield-integrations/DynamicRangeOrdersIntegrationDeployer.sol#382) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables
```

```
AccessControlEnumerableUpgradeable.grantRole(bytes32,address) (node_modules/@openzeppelin/contracts-upgradeable/access/AccessControlEnumerableUpgradeable.sol#69-72) ignores return value by _roleMembers[role].add(account) (node_modules/@openzeppelin/contracts-upgradeable/access/AccessControlEnumerableUpgradeable.sol#7
1)
AccessControlEnumerableUpgradeable.revokeRole(bytes32,address) (node_modules/@openzeppelin/contracts-upgradeable/access/AccessControlEnumerableUpgradeable.sol#77-80) ignores return value by _roleMembers[role].remove(account) (node_modules/@openzeppelin/contracts-upgradeable/access/AccessControlEnumerableUpgradeable.sol#79)
AccessControlEnumerableUpgradeable.renounceRole(bytes32,address) (node_modules/@openzeppelin/contracts-upgradeable/access/AccessControlEnumerableUpgradeable.sol#85-88) ignores return value by _roleMembers[role].remove(account) (node_modules/@openzeppelin/contracts-upgradeable/access/AccessControlEnumerableUpgradeable.sol#87)
AccessControlEnumerableUpgradeable._setupRole(bytes32,address) (node_modules/@openzeppelin/contracts-upgradeable/access/AccessControlEnumerableUpgradeable.sol#93-96) ignores return value by _roleMembers[role].add(account) (node_modules/@openzeppelin/contracts-upgradeable/access/AccessControlEnumerableUpgradeable.sol#9
5)
SushiSwapTrader.swapExactInput(address,address,address,uint256,uint256) (contracts/core/SushiSwapTrader.sol#125-179) ignores return value by ISushiSwapRouter(swapRouterAddress).swapExactTokensForTokens(amountIn,amountOutMin,path,recipient,deadline) (contracts/core/SushiSwapTrader.sol#157-170)
UniswapTrader.swapExactInput(address,address,address,uint256) (contracts/core/UniswapTrader.sol#286-361) ignores return value by IUniswapSwapRouter(swapRouterAddress).exactInput(params) (contracts/core/UniswapTrader.sol#312-358)
UniswapTrader.swapExactInput(address,address,address,uint256) (contracts/core/UniswapTrader.sol#286-361) ignores return value by IUniswapSwapRouter(swapRouterAddress).exactInputSingle(exactInputSingleParams) (contracts/core/UniswapTrader.sol#348-360)
UniswapTrader.swapExactOutput(address,address,address,uint256) (contracts/core/UniswapTrader.sol#368-440) ignores return value by IUniswapSwapRouter(swapRouterAddress).exactOutput(params) (contracts/core/UniswapTrader.sol#394-402)
UniswapTrader.swapExactOutput(address,address,address,uint256) (contracts/core/UniswapTrader.sol#368-440) ignores return value by IUniswapSwapRouter(swapRouterAddress).exactOutputSingle(exactOutputSingleParams) (contracts/core/UniswapTrader.sol#427-439)
YieldManager.processYield() (contracts/core/YieldManager.sol#266-315) ignores return value by IUniswapTrader(moduleMap.getModuleAddress(Modules.UniswapTrader)).swapExactInput(address(token),address(weth),address(this),token.balanceOf(moduleMap.getModuleAddress(Modules.UniswapTrader))) (contracts/core/YieldManager.sol
#295-301)
YieldManager.bioBuyBurn() (contracts/core/YieldManager.sol#397-412) ignores return value by ISushiSwapTrader(moduleMap.getModuleAddress(Modules.SushiSwapTrader)).bioBuyBurn() (contracts/core/YieldManager.sol#405-406)
AaveIntegration.withdraw(address,uint256) (contracts/yield-integrations/AaveIntegration.sol#52-85) ignores return value by IAaveLendingPool(lendingPoolAddress).withdraw(tokenAddress,amount,address(this)) (contracts/yield-integrations/AaveIntegration.sol#65-71)
AaveIntegration.harvestYield() (contracts/yield-integrations/AaveIntegration.sol#118-141) ignores return value by IAaveLendingPool(lendingPoolAddress).withdraw(tokenAddress,aTokenBalance - balances[tokenAddress],address(moduleMap.getModuleAddress(Modules.YieldManager))) (contracts/yield-integrations/AaveIntegration.s
ol#131-137)
DynamicRangeOrdersIntegration.increaseLiquidityPositions(uint256[]) (contracts/yield-integrations/DynamicRangeOrdersIntegration.sol#395-445) ignores return value by dynamicRangeOrdersIntegrationDeployer.increaseLiquidityPosition{liquidityPositions[liquidityPositionKeys[liquidityPositionIndex]].id,liquidityPositions[l
iquidityPositionKeys[liquidityPositionIndex]].token0,liquidityPositions[liquidityPositionKeys[liquidityPositionIndex]].token1,amount0Desired,amount1Desired} (contracts/yield-integrations/DynamicRangeOrdersIntegration.sol#415-424)
DynamicRangeOrdersIntegration.rerangeLiquidityPosition(bytes32) (contracts/yield-integrations/DynamicRangeOrdersIntegration.sol#476-518) ignores return value by dynamicRangeOrdersIntegrationDeployer.decreaseLiquidityPosition{liquidityPositionKey,positionActualBaseStableceivnValue} (contracts/yield-integrations/Dynamic
RangeOrdersIntegration.sol#483-486)
DynamicRangeOrdersIntegration.closeExcessLiquidityPositions() (contracts/yield-integrations/DynamicRangeOrdersIntegration.sol#521-547) ignores return value by dynamicRangeOrdersIntegrationDeployer.decreaseLiquidityPosition{liquidityPositionKeys[liquidityPositionIndex],positionActualBaseStableceivnValue - positionDesir
edBaseStableceivnValue} (contracts/yield-integrations/DynamicRangeOrdersIntegration.sol#543-544)
DynamicRangeOrdersIntegrationDeployer.swapExcessTokensForBaseStableceivn(uint256[]) (contracts/yield-integrations/DynamicRangeOrdersIntegrationDeployer.sol#85-128) ignores return value by uniswapTrader.swapExactInput(tokenAddress,baseStableceivnAddress,address(this),tokenBalance - tokenDesiredAmounts[tokenId]) (contrac
```

```
DynamicRangeOrdersIntegrationDeployer.swapExcessBaseStableCoinForTokens(uint256[]) (contracts/yield-integrations/DynamicRangeOrdersIntegrationDeployer.sol#123-173) ignores return value by uniswapTrader.swapExactInput(baseStableCoinAddress,tokenAddress,address(this),baseStableCoinAmount) (contracts/yield-integrations/DynamicRangeOrdersIntegrationDeployer.sol#154-169)
DynamicRangeOrdersIntegrationDeployer.mintLiquidityPosition(address,uint24,int24,int24,uint256,uint256) (contracts/yield-integrations/DynamicRangeOrdersIntegrationDeployer.sol#184-239) ignores return value by IUniswapPositionManager(positionManagerAddress).mint(mintParams) (contracts/yield-integrations/DynamicRangeOrdersIntegrationDeployer.sol#230-238)
DynamicRangeOrdersIntegrationDeployer.increaseLiquidityPosition(uint256,address,uint256,uint256) (contracts/yield-integrations/DynamicRangeOrdersIntegrationDeployer.sol#247-298) ignores return value by IUniswapPositionManager(positionManagerAddress).increaseLiquidity(increaseLiquidityParams) (contracts/yield-integrations/DynamicRangeOrdersIntegrationDeployer.sol#282-289)
DynamicRangeOrdersIntegrationDeployer.swapReservesForWithdrawalToken(address) (contracts/yield-integrations/DynamicRangeOrdersIntegrationDeployer.sol#376-484) ignores return value by IUniswapTrader(moduleMap.getModuleAddress(Modules.UniswapTrader)).swapExactInput(address(baseStableCoin),tokenAddress,address(this),baseStableCoin.balanceOf(moduleMap.getModuleAddress(Modules.UniswapTrader))) (contracts/yield-integrations/DynamicRangeOrdersIntegrationDeployer.sol#394-402)
DynamicRangeOrdersIntegrationDeployer.decreaseLiquidityPosition(bytes32,uint256) (contracts/yield-integrations/DynamicRangeOrdersIntegrationDeployer.sol#486-502) ignores return value by IUniswapPositionManager(positionManagerAddress).decreaseLiquidity(decreaseLiquidityParams) (contracts/yield-integrations/DynamicRangeOrdersIntegrationDeployer.sol#449-457)
DynamicRangeOrdersIntegrationDeployer.harvestYield(bytes32[]) (contracts/yield-integrations/DynamicRangeOrdersIntegrationDeployer.sol#485-618) ignores return value by IUniswapTrader(moduleMap.getModuleAddress(Modules.UniswapTrader)).swapExactInput(tokenAddress,baseStableCoinAddress,address(this),tokensHarvestedVale[tokenId]) (contracts/yield-integrations/DynamicRangeOrdersIntegrationDeployer.sol#578-576)
DynamicRangeOrdersIntegrationDeployer.harvestYield(bytes32[]) (contracts/yield-integrations/DynamicRangeOrdersIntegrationDeployer.sol#485-618) ignores return value by IUniswapTrader(moduleMap.getModuleAddress(Modules.UniswapTrader)).swapExactInput(baseStableCoinAddress,tokenAddress_scope_1,moduleMap.getModuleAddress(Modules.YieldManager),baseStableCoinAmount) (contracts/yield-integrations/DynamicRangeOrdersIntegrationDeployer.sol#602-608)
DynamicRangeOrdersIntegrationDeployer.collectTokensFromPosition(uint256,uint128,uint128) (contracts/yield-integrations/DynamicRangeOrdersIntegrationDeployer.sol#628-652) ignores return value by IUniswapPositionManager(positionManagerAddress).collect(collectParams) (contracts/yield-integrations/DynamicRangeOrdersIntegrationDeployer.sol#641-651)
SushiSwapIntegration.swapExactInput(address,address,address,uint256) (contracts/yield-integrations/SushiSwapIntegration.sol#383-398) ignores return value by IERC20MetadataUpgradeable(tokenIn).approve(swapRouterAddress,amountIn) (contracts/yield-integrations/SushiSwapIntegration.sol#395)
UniswapIntegration.increaseLiquidityPositions(uint256[]) (contracts/yield-integrations/UniswapIntegration.sol#485-564) ignores return value by uniswapIntegration.increaseLiquidityPosition(liquidityPositions[liquidityPositionIndex].id,liquidityPositions[liquidityPositionIndex].tokenA,liquidityPositions[liquidityPositionIndex].tokenB,liquidityPositions[liquidityPositionIndex].token1,amount0Desired,amount1Desired) (contracts/yield-integrations/UniswapIntegration.sol#424-430)
UniswapIntegration.closeExcessLiquidityPositions() (contracts/yield-integrations/UniswapIntegration.sol#467-483) ignores return value by uniswapIntegration.decreaseLiquidityPosition(liquidityPositionIndex,positionActualBaseStablecoinValue - positionDesiredBaseStablecoinValue) (contracts/yield-integrations/UniswapIntegration.sol#477-490)
UniswapIntegration.swapExcessTokensForBaseStablecoin(uint256[]) (contracts/yield-integrations/UniswapIntegration.sol#493-518) ignores return value by uniswapTrader.swapExactInput(tokenAddress,baseStableCoinAddress,address(this),tokenBalance - tokenDesiredAmounts[tokenId]) (contracts/yield-integrations/UniswapIntegration.sol#509-514)
UniswapIntegration.swapExcessBaseStablecoinForTokens(uint256[]) (contracts/yield-integrations/UniswapIntegration.sol#521-571) ignores return value by uniswapTrader.swapExactInput(baseStableCoinAddress,tokenAddress,address(this),baseStableCoinAmount) (contracts/yield-integrations/UniswapIntegration.sol#552-567)
UniswapIntegration.mintLiquidityPosition(address,address,uint24,int24,int24,uint256,uint256) (contracts/yield-integrations/UniswapIntegration.sol#582-637) ignores return value by IUniswapPositionManager(positionManagerAddress).mint(mintParams) (contracts/yield-integrations/UniswapIntegration.sol#620-636)
UniswapIntegration.increaseLiquidityPosition(uint256,address,address,uint256,uint256) (contracts/yield-integrations/UniswapIntegration.sol#245-288) ignores return value by IUniswapPositionManager(positionManagerAddress).increaseLiquidity(increaseLiquidityParams) (contracts/yield-integrations/UniswapIntegration.sol#279-287)
UniswapIntegration.swapReservesForWithdrawalToken(address) (contracts/yield-integrations/UniswapIntegration.sol#374-482) ignores return value by IUniswapTrader(moduleMap.getModuleAddress(Modules.UniswapTrader)).swapExactInput(address(baseStableCoin),tokenAddress,address(this),baseStableCoin.balanceOf(moduleMap.getModuleAddress(Modules.UniswapTrader))) (contracts/yield-integrations/UniswapIntegration.sol#392-400)
UniswapIntegration.decreaseLiquidityPosition(uint256,uint256) (contracts/yield-integrations/UniswapIntegration.sol#484-488) ignores return value by IUniswapPositionManager(positionManagerAddress).decreaseLiquidity(decreaseLiquidityParams) (contracts/yield-integrations/UniswapIntegration.sol#462-468)
UniswapIntegrationDeployer.harvestYield() (contracts/yield-integrations/UniswapIntegrationDeployer.sol#491-618) ignores return value by IUniswapTrader(moduleMap.getModuleAddress(Modules.UniswapTrader)).swapExactInput(tokenAddress,baseStableCoinAddress,address(this),tokensHarvestedVale[tokenId]) (contracts/yield-integrations/UniswapIntegrationDeployer.sol#578-576)
UniswapIntegrationDeployer.harvestYield() (contracts/yield-integrations/UniswapIntegrationDeployer.sol#491-618) ignores return value by IUniswapTrader(moduleMap.getModuleAddress(Modules.UniswapTrader)).swapExactInput(baseStableCoinAddress,tokenAddress_scope_1,moduleMap.getModuleAddress(Modules.YieldManager),baseStableCoinAmount) (contracts/yield-integrations/UniswapIntegrationDeployer.sol#602-608)
UniswapIntegrationDeployer.collectTokensFromPosition(uint256,uint128,uint128) (contracts/yield-integrations/UniswapIntegrationDeployer.sol#628-652) ignores return value by IUniswapPositionManager(positionManagerAddress).collect(collectParams) (contracts/yield-integrations/UniswapIntegrationDeployer.sol#641-651)
YearnIntegration.deploy() (contracts/yield-integrations/YearnIntegration.sol#181-324) ignores return value by IYearnVaultV2(vaultAddress).deposit(tokenAmount,address(this)) (contracts/yield-integrations/YearnIntegration.sol#121-123)
YearnIntegration.getVaultAddress(address) (contracts/yield-integrations/YearnIntegration.sol#181-187) ignores return value by IYearnRegistry(yearnRegistryAddress).latestVault(token) (contracts/yield-integrations/YearnIntegration.sol#182-186)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
```

```
IntegrationMap.initialize(address[],address,address,address).wethTokenAddress_ (contracts/core/IntegrationMap.sol#31) lacks a zero-check on :
        - wethTokenAddress = wethTokenAddress_ (contracts/core/IntegrationMap.sol#35)
IntegrationMap.initialize(address[],address,address,address).bisoTokenAddress_ (contracts/core/IntegrationMap.sol#32) lacks a zero-check on :
        - bisoTokenAddress = bisoTokenAddress_ (contracts/core/IntegrationMap.sol#36)
SushiSwapTrader.initialize(address[],address,address,address,uint24).factoryAddress_ (contracts/core/SushiSwapTrader.sol#54) lacks a zero-check on :
        - factoryAddress = factoryAddress_ (contracts/core/SushiSwapTrader.sol#64)
SushiSwapTrader.initialize(address[],address,address,address,uint24).swapRouterAddress_ (contracts/core/SushiSwapTrader.sol#55) lacks a zero-check on :
        - swapRouterAddress = swapRouterAddress_ (contracts/core/SushiSwapTrader.sol#65)
UniswapTrader.initialize(address[],address,address,address).factoryAddress_ (contracts/core/UniswapTrader.sol#74) lacks a zero-check on :
        - factoryAddress = factoryAddress_ (contracts/core/UniswapTrader.sol#78)
UniswapTrader.initialize(address[],address,address,address).swapRouterAddress_ (contracts/core/UniswapTrader.sol#75) lacks a zero-check on :
        - swapRouterAddress = swapRouterAddress_ (contracts/core/UniswapTrader.sol#80)
YieldManager.initialize(address[],address,uint256,uint32,uint32,uint32,address,address).gasAccount_ (contracts/core/YieldManager.sol#68) lacks a zero-check on :
        - gasAccount = gasAccount_ (contracts/core/YieldManager.sol#78)
YieldManager.initialize(address[],address,uint256,uint32,uint32,uint32,address,address).treasuryAccount_ (contracts/core/YieldManager.sol#61) lacks a zero-check on :
        - treasuryAccount = treasuryAccount_ (contracts/core/YieldManager.sol#71)
YieldManager.updateGasAccount(address).gasAccount_ (contracts/core/YieldManager.sol#101) lacks a zero-check on :
        - gasAccount = gasAccount_ (contracts/core/YieldManager.sol#109)
YieldManager.updateTreasuryAccount(address).treasuryAccount_ (contracts/core/YieldManager.sol#113) lacks a zero-check on :
        - treasuryAccount = treasuryAccount_ (contracts/core/YieldManager.sol#120)
AaveIntegration.initialize(address[],address,address).lendingPoolAddress_ (contracts/yield-integrations/AaveIntegration.sol#32) lacks a zero-check on :
        - lendingPoolAddress = lendingPoolAddress_ (contracts/yield-integrations/AaveIntegration.sol#36)
DynamicRangeOrdersIntegration.initialize(address[],address,address,address).factoryAddress_ (contracts/yield-integrations/DynamicRangeOrdersIntegration.sol#63) lacks a zero-check on :
        - factoryAddress = factoryAddress_ (contracts/yield-integrations/DynamicRangeOrdersIntegration.sol#68)
DynamicRangeOrdersIntegration.initialize(address[],address,address,address).positionManagerAddress_ (contracts/yield-integrations/DynamicRangeOrdersIntegration.sol#64) lacks a zero-check on :
        - positionManagerAddress = positionManagerAddress_ (contracts/yield-integrations/DynamicRangeOrdersIntegration.sol#69)
DynamicRangeOrdersIntegrationDeployer.initialize(address[],address,address,address).factoryAddress_ (contracts/yield-integrations/DynamicRangeOrdersIntegrationDeployer.sol#46) lacks a zero-check on :
        - factoryAddress = factoryAddress_ (contracts/yield-integrations/DynamicRangeOrdersIntegrationDeployer.sol#50)
DynamicRangeOrdersIntentionDeployer.initialize(address[],address,address,address).positionManagerAddress_ (contracts/yield-integrations/DynamicRangeOrdersIntegrationDeployer.sol#45) lacks a zero-check on :
        - positionManagerAddress = positionManagerAddress_ (contracts/yield-integrations/DynamicRangeOrdersIntegrationDeployer.sol#51)
SushiSwapIntegration.initialize(address[],address,address,address,address,uint24).factoryAddress_ (contracts/yield-integrations/SushiSwapIntegration.sol#52) lacks a zero-check on :
        - factoryAddress = factoryAddress_ (contracts/yield-integrations/SushiSwapIntegration.sol#59)
SushiSwapIntegration.initialize(address[],address,address,address,address,uint24).swapRouterAddress_ (contracts/yield-integrations/SushiSwapIntegration.sol#57) lacks a zero-check on :
        - swapRouterAddress = swapRouterAddress_ (contracts/yield-integrations/SushiSwapIntegration.sol#60)
SushiSwapIntegration.initialize(address[],address,address,address,address,uint24).masterChef_ (contracts/yield-integrations/SushiSwapIntegration.sol#57) lacks a zero-check on :
        - masterChef = masterChef_ (contracts/yield-integrations/SushiSwapIntegration.sol#61)
SushiSwapIntegration.initialize(address[],address,address,address,address,uint24).sushi_ (contracts/yield-integrations/SushiSwapIntegration.sol#57) lacks a zero-check on :
        - sushi = sushi_ (contracts/yield-integrations/SushiSwapIntegration.sol#63)
UniswapIntegration.initialize(address[],address,address,address).factoryAddress_ (contracts/yield-integrations/UniswapIntegration.sol#73) lacks a zero-check on :
        - factoryAddress = factoryAddress_ (contracts/yield-integrations/UniswapIntegration.sol#78)
UniswapIntegration.initialize(address[],address,address,address).positionManagerAddress_ (contracts/yield-integrations/UniswapIntegration.sol#74) lacks a zero-check on :
        - positionManagerAddress = positionManagerAddress_ (contracts/yield-integrations/UniswapIntegration.sol#79)
UniswapIntegrationDeployer.initialize(address[],address,address,address).factoryAddress_ (contracts/yield-integrations/UniswapIntegrationDeployer.sol#44) lacks a zero-check on :
        - factoryAddress = factoryAddress_ (contracts/yield-integrations/UniswapIntegrationDeployer.sol#50)
UniswapIntegrationDeployer.initialize(address[],address,address,address).positionManagerAddress_ (contracts/yield-integrations/UniswapIntegrationDeployer.sol#45) lacks a zero-check on :
        - positionManagerAddress = positionManagerAddress_ (contracts/yield-integrations/UniswapIntegrationDeployer.sol#51)
YearnIntegration.initialize(address[],address,address).yearnRegistryAddress_ (contracts/yield-integrations/YearnIntegration.sol#32) lacks a zero-check on :
        - yearnRegistryAddress = yearnRegistryAddress_ (contracts/yield-integrations/YearnIntegration.sol#36)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
```

```
BisoRewards.getUserBisoRewards(address) (contracts/core/BisoRewards.sol#141-160) has external calls inside a loop: tokenId < integrationMap.getTokenAddressesLength() (contracts/core/BisoRewards.sol#154-156)
BisoRewards.getUserBisoRewards(address) (contracts/core/BisoRewards.sol#141-160) has external calls inside a loop: userBisoRewards += stored[integrationMap.getTokenAddress(tokenId),account] (contracts/core/BisoRewards.sol#156-160)
EtherRewards.claimEthRewards(address) (contracts/core/EtherRewards.sol#76-90) has external calls inside a loop: ! IIntegration(mapIntegration(mapI).getTokenAddress(tokenId) (contracts/core/EtherRewards.sol#87)
EtherRewards.getUserEthRewards(address) (contracts/core/EtherRewards.sol#132-140) has external calls inside a loop: token = IIntegration(mapIntegration(mapI).getTokenAddress(tokenId) (contracts/core/EtherRewards.sol#141)
StrategyMap.updateIntegrations(uint256,IStrategyMap.WeightedIntegration[]) (contracts/core/StrategyMap.sol#123-195) has external calls inside a loop: token = integrationMap.getTokenAddress(i_scope_1) (contracts/core/StrategyMap.sol#164)
StrategyMap.deleteStrategy(uint256) (contracts/core/StrategyMap.sol#195-220) has external calls inside a loop: require(bool,string)(getStrategyTokenBalance(id,integrationMap.getTokenAddress(i)) == 0,Strategy in use) (contracts/core/StrategyMap.sol#204-207)
StrategyMap.enterStrategy(uint256,address,address[],uint256[]) (contracts/core/StrategyMap.sol#276-322) has external calls inside a loop: require(bool,string)(isIntegrationMap.getTokenAcceptingDeposits(tokens[i]),Token unavailable) (contracts/core/StrategyMap.sol#294-297)
StrategyMap.enterStrategy(uint256,address,address[],uint256[]) (contracts/core/StrategyMap.sol#276-322) has external calls inside a loop: require(bool,string)(userPositions.userTokenBalance(tokens[i],user) >= amounts[i],User lacks funds) (contracts/core/StrategyMap.sol#300-301)
StrategyMap.exitStrategy(uint256,address,address[],uint256[]) (contracts/core/StrategyMap.sol#324-343) has external calls inside a loop: require(bool,string)(integrationMap.getTokenAcceptingWithdrawals(tokens[i]),Token unavailable) (contracts/core/StrategyMap.sol#331-334)
UserPositions.deposit(address,address[],uint256[],uint256) (contracts/core/UserPositions.sol#98-160) has external calls inside a loop: require(bool,string)(integrationMap.getTokenAcceptingDeposits(tokens[tokenId]),UserPositions::deposit: This token is not accepting deposits) (contracts/core/UserPositions.sol#110-113)
UserPositions.deposit(address,address[],uint256[],uint256) (contracts/core/UserPositions.sol#98-160) has external calls inside a loop: beforeBalance = IERC20MetadataUpgradeable.balanceOf(moduleMap.getModuleAddress(Modules.Kernel)) (contracts/core/UserPositions.sol#120-123)
UserPositions.deposit(address,address[],uint256[],uint256) (contracts/core/UserPositions.sol#98-160) has external calls inside a loop: erc20.safeTransferFrom(depositor,moduleMap.getModuleAddress(Modules.Kernel),amounts[tokenId]) (contracts/core/UserPositions.sol#130-134)
UserPositions.deposit(address,address[],uint256[],uint256) (contracts/core/UserPositions.sol#98-160) has external calls inside a loop: afterBalance = erc20.balanceOf(moduleMap.getModuleAddress(Modules.Kernel)) (contracts/core/UserPositions.sol#137-139)
UserPositions.deposit(address,address[],uint256[],uint256) (contracts/core/UserPositions.sol#98-160) has external calls inside a loop: IBisoRewards(moduleMap.getModuleAddress(Modules.BisoRewards)).increaseRewards(tokens[tokenId],depositor,actualAmount) (contracts/core/UserPositions.sol#143-144)
UserPositions.deposit(address,address[],uint256[],uint256) (contracts/core/UserPositions.sol#98-160) has external calls inside a loop: IEtherRewards(moduleMap.getModuleAddress(Modules.EtherRewards)).updateUserRewards(tokens[tokenId],depositor) (contracts/core/UserPositions.sol#155-156)
UserPositions._withdraw(address,address[],uint256[],bool) (contracts/core/UserPositions.sol#213-306) has external calls inside a loop: require(bool,string)(integrationMap.getTokenAcceptingWithdrawals(tokens[tokenId]),UserPositions._withdraw: This token is not accepting withdrawals) (contracts/core/UserPositions.sol#254-257)
UserPositions._withdraw(address,address[],uint256[],bool) (contracts/core/UserPositions.sol#213-306) has external calls inside a loop: IERC20MetadataUpgradeable(tokens[tokenId]).balanceOf(moduleMap.getModuleAddress(Modules.Kernel)) < amounts[tokenId] (contracts/core/UserPositions.sol#256-266)
UserPositions._withdraw(address,address[],uint256[],bool) (contracts/core/UserPositions.sol#213-306) has external calls inside a loop: IERC20MetadataUpgradeable(tokens[tokenId]).balanceOf(moduleMap.getModuleAddress(Modules.Kernel)) < amounts[tokenId] (contracts/core/UserPositions.sol#272-274)
UserPositions._withdraw(address,address[],uint256[],bool) (contracts/core/UserPositions.sol#213-306) has external calls inside a loop: amounts[tokenId] = IERC20MetadataUpgradeable(tokens[tokenId]).balanceOf(moduleMap.getModuleAddress(Modules.Kernel)) (contracts/core/UserPositions.sol#277-278)
UserPositions._withdraw(address,address[],uint256[],bool) (contracts/core/UserPositions.sol#213-306) has external calls inside a loop: IBisoRewards(moduleMap.getModuleAddress(Modules.BisoRewards)).decreaseRewards(tokens[tokenId],recipient,amounts[tokenId]) (contracts/core/UserPositions.sol#294-295)
UserPositions._withdraw(address,address[],uint256[],bool) (contracts/core/UserPositions.sol#213-306) has external calls inside a loop: IEtherRewards(moduleMap.getModuleAddress(Modules.EtherRewards)).updateUserRewards(tokens[tokenId],recipient) (contracts/core/UserPositions.sol#297-298)
UserPositions._withdraw(address,address[],uint256[],bool) (contracts/core/UserPositions.sol#213-306) has external calls inside a loop: IERC20MetadataUpgradeable(tokens[tokenId]).safeTransferFrom(moduleMap.getModuleAddress(Modules.Kernel),recipient,amounts[tokenId]) (contracts/core/UserPositions.sol#286-290)
UserPositions.closePositionsForWithdrawal(address,uint256) (contracts/core/UserPositions.sol#308-351) has external calls inside a loop: integrationId < integrationMap.getIntegrationAddressesLength() (contracts/core/UserPositions.sol#321)
UserPositions.closePositionsForWithdrawal(address,uint256) (contracts/core/UserPositions.sol#308-351) has external calls inside a loop: integrationAddress = integrationMap.getIntegrationAddress(integrationId) (contracts/core/UserPositions.sol#324-326)
UserPositions.closePositionsForWithdrawal(address,uint256) (contracts/core/UserPositions.sol#308-351) has external calls inside a loop: desiredWithdrawAmount = (amount * strategyMap.getIntegrationWeight(integrationAddress)) / integrationWeightSum (contracts/core/UserPositions.sol#327-329)
UserPositions.closePositionsForWithdrawal(address,uint256) (contracts/core/UserPositions.sol#308-351) has external calls inside a loop: desiredWithdrawAmount > IIntegration(integrationAddress).getBalance(token) (contracts/core/UserPositions.sol#332-333)
UserPositions.closePositionsForWithdrawal(address,uint256) (contracts/core/UserPositions.sol#308-351) has external calls inside a loop: desiredWithdrawAmount = IIntegration(integrationAddress).getBalance(token) (contracts/core/UserPositions.sol#335-337)
UserPositions.closePositionsForWithdrawal(address,uint256) (contracts/core/UserPositions.sol#308-351) has external calls inside a loop: IIntegration(integrationAddress).withdraw(token,desiredWithdrawAmount) (contracts/core/UserPositions.sol#340)
UserPositions.fullyClosePositionsForWithdrawal(address,uint256) (contracts/core/UserPositions.sol#355-386) has external calls inside a loop: integration = IIntegration(integrationMap.getIntegrationAddress(integrationId)) (contracts/core/UserPositions.sol#365-367)
UserPositions.fullyClosePositionsForWithdrawal(address,uint256) (contracts/core/UserPositions.sol#355-386) has external calls inside a loop: integration.withdraw(token,integration.getBalance(token)) (contracts/core/UserPositions.sol#370)
UserPositions.fullyClosePositionsForWithdrawal(address,uint256) (contracts/core/UserPositions.sol#355-386) has external calls inside a loop: integrationId == integrationMap.getIntegrationAddressesLength() - 1 || IERC20MetadataUpgradeable(token).balanceOf(moduleMap.getModuleAddress(Modules.Kernel)) >= amount (contracts/core/UserPositions.sol#373-377)
UserPositions.transferToStrategy(address,address[],uint256[]) (contracts/core/UserPositions.sol#394-428) has external calls inside a loop: IBisoRewards(moduleMap.getModuleAddress(Modules.BisoRewards)).decreaseRewards(tokens[tokenId],recipient,amounts[tokenId]) (contracts/core/UserPositions.sol#418-419)
UserPositions.transferToStrategy(address,address[],uint256[]) (contracts/core/UserPositions.sol#394-428) has external calls inside a loop: IEtherRewards(moduleMap.getModuleAddress(Modules.EtherRewards)).updateUserRewards(tokens[tokenId],recipient) (contracts/core/UserPositions.sol#421-422)
UserPositions.transferFromStrategy(address,address[],uint256[]) (contracts/core/UserPositions.sol#436-465) has external calls inside a loop: require(bool,string)(integrationMap.getTokenAcceptingDeposits(tokens[tokenId]),UserPositions::deposit: This token is not accepting deposits) (contracts/core/UserPositions.sol#446-449)
UserPositions.transferFromStrategy(address,address[],uint256[]) (contracts/core/UserPositions.sol#436-465) has external calls inside a loop: IBisoRewards(moduleMap.getModuleAddress(Modules.BisoRewards)).increaseRewards(tokens[tokenId],recipient,amounts[tokenId]) (contracts/core/UserPositions.sol#456-457)
UserPositions.transferFromStrategy(address,address[],uint256[]) (contracts/core/UserPositions.sol#436-465) has external calls inside a loop: IEtherRewards(moduleMap.getModuleAddress(Modules.EtherRewards)).updateUserRewards(tokens[tokenId],recipient) (contracts/core/UserPositions.sol#458-459)
UserPositions._increaseBisoRewards(address,uint256) (contracts/core/UserPositions.sol#473-506) has external calls inside a loop: token = integrationMap.getTokenAddress(tokenId) (contracts/core/UserPositions.sol#494)
UserPositions._increaseBisoRewards() (contracts/core/UserPositions.sol#473-506) has external calls inside a loop: tokenBisoRewardWeight = integrationMap.getTokenBisoRewardWeight(token) (contracts/core/UserPositions.sol#499-501)
UserPositions._claimBisoRewards(address) (contracts/core/UserPositions.sol#561-590) has external calls inside a loop: token = integrationMap.getTokenAddress(tokenId) (contracts/core/UserPositions.sol#573)
UserPositions._claimBisoRewards(address) (contracts/core/UserPositions.sol#561-590) has external calls inside a loop: bisoRewards.earned(token,recipient) > 0 (contracts/core/UserPositions.sol#577)
UserPositions._claimBisoRewards(address) (contracts/core/UserPositions.sol#561-590) has external calls inside a loop: bisoClaimed += IBisoRewards(moduleMap.getModuleAddress(Modules.BisoRewards)).claimReward(token,recipient) (contracts/core/UserPositions.sol#578-588)
YieldManager._rebalance() (contracts/core/YieldManager.sol#148-204) has external calls inside a loop: integration = integrationMap.getIntegrationAddress(i) (contracts/core/YieldManager.sol#152)
YieldManager._rebalance() (contracts/core/YieldManager.sol#148-204) has external calls inside a loop: token = integrationMap.getTokenAddress(i) (contracts/core/YieldManager.sol#154)
YieldManager._rebalance() (contracts/core/YieldManager.sol#148-204) has external calls inside a loop: numerator = integrationMap.getTokenAllocPercentNumerator(token) (contracts/core/YieldManager.sol#155)
YieldManager._rebalance() (contracts/core/YieldManager.sol#148-204) has external calls inside a loop: grossAmountInvested = strategyMap.getExpectedBalance(integration,token) (contracts/core/YieldManager.sol#157-160)
YieldManager._rebalance() (contracts/core/YieldManager.sol#148-204) has external calls inside a loop: actualBalance = IIntegration(integration).getBalance(token) (contracts/core/YieldManager.sol#161)
YieldManager._rebalance() (contracts/core/YieldManager.sol#148-204) has external calls inside a loop: IERC20MetadataUpgradeable(token).balanceOf(moduleMap.getModuleAddress(Modules.Kernel)) >= shortage (contracts/core/YieldManager.sol#172-174)
YieldManager._rebalance() (contracts/core/YieldManager.sol#148-204) has external calls inside a loop: balanceBefore = IERC20MetadataUpgradeable(token).balanceOf(integration) (contracts/core/YieldManager.sol#176-178)
YieldManager._rebalance() (contracts/core/YieldManager.sol#148-204) has external calls inside a loop: IERC20MetadataUpgradeable(token).safeTransferFrom(moduleMap.getModuleAddress(Modules.Kernel),integration,shortage) (contracts/core/YieldManager.sol#179-183)
YieldManager._rebalance() (contracts/core/YieldManager.sol#148-204) has external calls inside a loop: balanceAfter = IERC20MetadataUpgradeable(token).balanceOf(integration) (contracts/core/YieldManager.sol#184-186)
YieldManager._rebalance() (contracts/core/YieldManager.sol#148-204) has external calls inside a loop: IIntegration(integration).deposit(token,balanceAfter - balanceBefore) (contracts/core/YieldManager.sol#188-191)
YieldManager._rebalance() (contracts/core/YieldManager.sol#148-204) has external calls inside a loop: IIntegration(integration).withdraw(token,actualBalance - desiredBalance) (contracts/core/YieldManager.sol#196-199)
YieldManager.harvestYield() (contracts/core/YieldManager.sol#216-265) has external calls inside a loop: IIntegration(integrationMap.getIntegrationAddress(integrationId)).harvestYield() (contracts/core/YieldManager.sol#220-229)
YieldManager.harvestYield() (contracts/core/YieldManager.sol#216-265) has external calls inside a loop: token = IERC20MetadataUpgradeable(integrationMap.getTokenAddress(tokenId)) (contracts/core/YieldManager.sol#233-235)
YieldManager.harvestYield() (contracts/core/YieldManager.sol#216-265) has external calls inside a loop: harvestedTokenAmount = token.balanceOf(address(this)) (contracts/core/YieldManager.sol#242)
```

```
Variable `DynamicRangeOrdersIntegration.increaseLiquidityPositions(uint256[]).amount1Desired (contracts/yield-integrations/DynamicRangeOrdersIntegration.sol#409)` in DynamicRangeOrdersIntegration.increaseLiquidityPositions(uint256[]) (contracts/yield-integrations/DynamicRangeOrdersIntegration.sol#395-465) potentially
 used before declaration: (amount0Desired,amount1Desired) = getIncreaseLiquidityAmounts(liquidityPositionKeys[liquidityPositionIndex],liquidityPositionDesiredDeployAmounts[liquidityPositionIndex]) (contracts/yield-integrations/DynamicRangeOrdersIntegration.sol#428-434)
Variable `DynamicRangeOrdersIntegration.increaseLiquidityPositions(uint256[]).amount0Desired (contracts/yield-integrations/DynamicRangeOrdersIntegration.sol#408)` in DynamicRangeOrdersIntegration.increaseLiquidityPositions(uint256[]) (contracts/yield-integrations/DynamicRangeOrdersIntegration.sol#395-465) potentially
 used before declaration: (amount0Desired,amount1Desired) = getIncreaseLiquidityAmounts(liquidityPositionKeys[liquidityPositionIndex],liquidityPositionDesiredDeployAmounts[liquidityPositionIndex]) (contracts/yield-integrations/DynamicRangeOrdersIntegration.sol#428-434)
Variable `DynamicRangeOrdersIntegrationDeployer.mintLiquidityPosition(address,address,uint24,int24,int24,uint256,uint256).returnedLiquidityPositionId (contracts/yield-integrations/DynamicRangeOrdersIntegrationDeployer.sol#232)` in DynamicRangeOrdersIntegrationDeployer.mintLiquidityPosition(address,address,uint24,int2
4,int24,uint256,uint256) (contracts/yield-integrations/DynamicRangeOrdersIntegrationDeployer.sol#184-229) potentially used before declaration: liquidityPositionId = returnedLiquidityPositionId (contracts/yield-integrations/DynamicRangeOrdersIntegrationDeployer.sol#226)
Variable `DynamicRangeOrdersIntegrationDeployer.collectTokensFromPosition(uint256,uint128,uint128).returnedAmount0 (contracts/yield-integrations/DynamicRangeOrdersIntegrationDeployer.sol#543)` in DynamicRangeOrdersIntegrationDeployer.collectTokensFromPosition(uint256,uint128,uint128) (contracts/yield-integrations/Dyn
amicRangeOrdersIntegrationDeployer.sol#528-652) potentially used before declaration: amount0Collected = returnedAmount0 (contracts/yield-integrations/DynamicRangeOrdersIntegrationDeployer.sol#643)
Variable `DynamicRangeOrdersIntegrationDeployer.collectTokensFromPosition(uint256,uint128,uint128).returnedAmount1 (contracts/yield-integrations/DynamicRangeOrdersIntegrationDeployer.sol#543)` in DynamicRangeOrdersIntegrationDeployer.collectTokensFromPosition(uint256,uint128,uint128) (contracts/yield-integrations/Dyn
amicRangeOrdersIntegrationDeployer.sol#528-652) potentially used before declaration: amount1Collected = returnedAmount1 (contracts/yield-integrations/DynamicRangeOrdersIntegrationDeployer.sol#644)
Variable `UniswapIntegration.increaseLiquidityPositions(uint256[]).amount0Desired (contracts/yield-integrations/UniswapIntegration.sol#417)` in UniswapIntegration.increaseLiquidityPositions(uint256[]) (contracts/yield-integrations/UniswapIntegration.sol#485-564) potentially used before declaration: (amount0Desired,am
ount1Desired) = getIncreaseLiquidityAmounts(liquidityPositionIndex,liquidityPositionDesiredDeployAmounts[liquidityPositionIndex]) (contracts/yield-integrations/UniswapIntegration.sol#534-540)
Variable `UniswapIntegration.increaseLiquidityPositions(uint256[]).amount1Desired (contracts/yield-integrations/UniswapIntegration.sol#418)` in UniswapIntegration.increaseLiquidityPositions(uint256[]) (contracts/yield-integrations/UniswapIntegration.sol#485-564) potentially used before declaration: (amount0Desired,am
ount1Desired) = getIncreaseLiquidityAmounts(liquidityPositionIndex,liquidityPositionDesiredDeployAmounts[liquidityPositionIndex]) (contracts/yield-integrations/UniswapIntegration.sol#534-540)
Variable `UniswapIntegrationDeployer.mintLiquidityPosition(address,address,uint24,int24,int24,uint256,uint256).returnedLiquidityPositionId (contracts/yield-integrations/UniswapIntegrationDeployer.sol#239)` in UniswapIntegrationDeployer.mintLiquidityPosition(address,address,uint24,int24,int24,uint256,uint256) (contrac
ts/yield-integrations/UniswapIntegrationDeployer.sol#182-237) potentially used before declaration: liquidityPositionId = returnedLiquidityPositionId (contracts/yield-integrations/UniswapIntegrationDeployer.sol#232)
Variable `UniswapIntegrationDeployer.collectTokensFromPosition(uint256,uint128,uint128).returnedAmount0 (contracts/yield-integrations/UniswapIntegrationDeployer.sol#643)` in UniswapIntegrationDeployer.collectTokensFromPosition(uint256,uint128,uint128) (contracts/yield-integrations/UniswapIntegrationDeployer.sol#628-6
52) potentially used before declaration: amount0Collected = returnedAmount0 (contracts/yield-integrations/UniswapIntegrationDeployer.sol#643)
Variable `UniswapIntegrationDeployer.collectTokensFromPosition(uint256,uint128,uint128).returnedAmount1 (contracts/yield-integrations/UniswapIntegrationDeployer.sol#643)` in UniswapIntegrationDeployer.collectTokensFromPosition(uint256,uint128,uint128) (contracts/yield-integrations/UniswapIntegrationDeployer.sol#628-6
52) potentially used before declaration: amount1Collected = returnedAmount1 (contracts/yield-integrations/UniswapIntegrationDeployer.sol#644)
Variable `YearnIntegration.getVaultAddress(address).vaultAddress (contracts/yield-integrations/YearnIntegration.sol#107)` in YearnIntegration.getVaultAddress(address) (contracts/yield-integrations/YearnIntegration.sol#181-187) potentially used before declaration: vaultAddress (contracts/yield-integrations/YearnInteg
ration.sol#183-184)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#pre-declaration-usage-of-local-variables

Reentrancy in SushiSwapIntegration._deploy(uint256,bytes) (contracts/yield-integrations/SushiSwapIntegration.sol#171-233):
        External calls:
        - IWethI(IntegrationMap(moduleMap).getModuleAddress(Modules.IntegrationMap)).getWethTokenAddress().withdraw(amount) (contracts/yield-integrations/SushiSwapIntegration.sol#202)
        - amounts = swapExactETHForTokens(amount / 2,(amountTokenDesired - (amountTokenDesired * 1 / 100)) / decimalsSubtractionExponent),wethAddress,getTokenToAdd(innerPoolInfo.tokenPair.token0,innerPoolInfo.tokenPair.token1),address(this)) (contracts/yield-integrations/SushiSwapIntegration.sol#186-192)
                - ISushiSwapRouter(swapRouterAddress).swapExactETHForTokens{value: amountWei}(amountOutMin,path,to,deadline) (contracts/yield-integrations/SushiSwapIntegration.sol#376)
        - IERC20MetadataUpgradeable(getTokenToAdd(innerPoolInfo.tokenPair.token0,innerPoolInfo.tokenPair.token1)).safeTransfer(moduleMap.getModuleAddress(Modules.Kernel),IERC20MetadataUpgradeable(getTokenToAdd(innerPoolInfo.tokenPair.token0,innerPoolInfo.tokenPair.token1)).balanceOf(address(this))) (contracts/yield-i
ntegrations/SushiSwapIntegration.sol#210)
        - IWethI(IntegrationMap(moduleMap).getModuleAddress(Modules.IntegrationMap)).getWethTokenAddress().deposit{value: ethBalance}() (contracts/yield-integrations/SushiSwapIntegration.sol#223)
        - IERC20MetadataUpgradeable(IIntegrationMap(moduleMap).getModuleAddress(Modules.IntegrationMap)).getWethTokenAddress()).safeTransfer(moduleMap.getModuleAddress(Modules.Kernel),ethBalance) (contracts/yield-integrations/SushiSwapIntegration.sol#224)
        - IERC20MetadataUpgradeable(address(innerPoolInfo.poolInfo.lpToken)).safeApprove(masterChef,liquidity) (contracts/yield-integrations/SushiSwapIntegration.sol#229)
        - ISushiSwapMasterChef(masterChef).deposit(pid,liquidity) (contracts/yield-integrations/SushiSwapIntegration.sol#230)
        - IERC20MetadataUpgradeable(getTokenToAdd(innerPoolInfo.tokenPair.token0,innerPoolInfo.tokenPair.token1)).safeApprove(swapRouterAddress,IERC20MetadataUpgradeable(getTokenToAdd(innerPoolInfo.tokenPair.token0,innerPoolInfo.tokenPair.token1)).allowance(address(this),swapRouterAddress) + (amountTokenDesired / dec
imalsSubtractionExponent)) (contracts/yield-integrations/SushiSwapIntegration.sol#203)
        - IERC20MetadataUpgradeable(getTokenToAdd(innerPoolInfo.tokenPair.token0,innerPoolInfo.tokenPair.token1)).safeApprove(swapRouterAddress,IERC20MetadataUpgradeable(getTokenToAdd(innerPoolInfo.tokenPair.token0,innerPoolInfo.tokenPair.token1)).allowance(address(this),swapRouterAddress) + (tokensReceived)) (contra
cts/yield-integrations/SushiSwapIntegration.sol#203)
        External calls sending eth:
        - amounts = swapExactETHForTokens(amount / 2,(amountTokenDesired - (amountTokenDesired * 1 / 100)) / decimalsSubtractionExponent),wethAddress,getTokenToAdd(innerPoolInfo.tokenPair.token0,innerPoolInfo.tokenPair.token1),address(this)) (contracts/yield-integrations/SushiSwapIntegration.sol#186-192)
                - ISushiSwapRouter(swapRouterAddress).swapExactETHForTokens{value: amountWei}(amountOutMin,path,to,deadline) (contracts/yield-integrations/SushiSwapIntegration.sol#376)
        - (liquidity) = ISushiSwapRouter(swapRouterAddress).addLiquidityETH{value: amount / 2}(getTokenToAdd(innerPoolInfo.tokenPair.token0,innerPoolInfo.tokenPair.token1),(amountTokenDesired / decimalsSubtractionExponent) - diff,(amountTokenMin / decimalsSubtractionExponent),amountWeiMin,address(this),block.timestam
p + 360) (contracts/yield-integrations/SushiSwapIntegration.sol#207-214)
        - IWethI(IntegrationMap(moduleMap).getModuleAddress(Modules.IntegrationMap)).getWethTokenAddress().deposit{value: ethBalance}() (contracts/yield-integrations/SushiSwapIntegration.sol#223)
        State variables written after the call(s):
        - poolBalances[pid] += amount (contracts/yield-integrations/SushiSwapIntegration.sol#222)
Reentrancy in UserPositions._withdraw(address,address[],uint256[],bool) (contracts/core/UserPositions.sol#233-304):
        External calls:
        - closePositionsForWithdrawal(tokens[tokenId],amounts[tokenId]) (contracts/core/UserPositions.sol#269)
                - integration.withdraw(token,integration.getBalance(token)) (contracts/core/UserPositions.sol#370)
                - IIntegration(integrationAddress).withdraw(token,desiredWithdrawAmount) (contracts/core/UserPositions.sol#348)
        - IERC20MetadataUpgradeable(tokens[tokenId]).safeTransferFrom(moduleMap.getModuleAddress(Modules.Kernel),recipient,amounts[tokenId]) (contracts/core/UserPositions.sol#286-290)
        - IBlosRewards(moduleMap.getModuleAddress(Modules.BlosRewards)).decreaseRewards(tokens[tokenId],recipient,amounts[tokenId]) (contracts/core/UserPositions.sol#294-295)
        - IEtherRewards(moduleMap.getModuleAddress(Modules.EtherRewards)).updateUserRewards(tokens[tokenId],recipient) (contracts/core/UserPositions.sol#297-298)
        State variables written after the call(s):
        - _totalSupply[tokens[tokenId]] -= amounts[tokenId] (contracts/core/UserPositions.sol#301)
Reentrancy in DynamicRangeOrdersIntegration._withdraw(address,uint256) (contracts/yield-integrations/DynamicRangeOrdersIntegration.sol#345-355):
        External calls:
        - IERC20MetadataUpgradeable(token).safeTransferFrom(address(dynamicRangeOrdersIntegrationDeployer),moduleMap.getModuleAddress(Modules.Kernel),amount) (contracts/yield-integrations/DynamicRangeOrdersIntegration.sol#347-351)
        State variables written after the call(s):
        - balances[tokenAddress] -= (balanceBefore - token.balanceOf(address(dynamicRangeOrdersIntegrationDeployer))) (contracts/yield-integrations/DynamicRangeOrdersIntegration.sol#353-354)
Reentrancy in UniswapIntegration._withdraw(address,uint256) (contracts/yield-integrations/UniswapIntegration.sol#353-367):
        External calls:
        - IERC20MetadataUpgradeable(token).safeTransferFrom(address(uniswapIntegrationDeployer),moduleMap.getModuleAddress(Modules.Kernel),amount) (contracts/yield-integrations/UniswapIntegration.sol#359-363)
        State variables written after the call(s):
        - balances[tokenAddress] -= (balanceBefore - token.balanceOf(address(uniswapIntegrationDeployer))) (contracts/yield-integrations/UniswapIntegration.sol#365-366)
Reentrancy in SushiSwapIntegration.addPool(uint256,uint256) (contracts/yield-integrations/SushiSwapIntegration.sol#67-83):
        External calls:
        - poolInfo = ISushiSwapMasterChef(masterChef).poolInfo(pid) (contracts/yield-integrations/SushiSwapIntegration.sol#69)
        State variables written after the call(s):
        - poolIds.push(pid) (contracts/yield-integrations/SushiSwapIntegration.sol#80)
        - pool[pid].poolInfo = poolInfo (contracts/yield-integrations/SushiSwapIntegration.sol#74)
        - pool[pid].added = true (contracts/yield-integrations/SushiSwapIntegration.sol#75)
        - pool[pid].weight = weight (contracts/yield-integrations/SushiSwapIntegration.sol#76)
        - pool[pid].tokenPair.token0 = ISushiSwapPair(address(poolInfo.lpToken)).token0() (contracts/yield-integrations/SushiSwapIntegration.sol#77)
        - pool[pid].tokenPair.token1 = ISushiSwapPair(address(poolInfo.lpToken)).token1() (contracts/yield-integrations/SushiSwapIntegration.sol#78)
Reentrancy in Kernel.bisoBuyBack() (contracts/core/Kernel.sol#597-599):
        External calls:
        - IYieldManager(moduleMap.getModuleAddress(Modules.YieldManager)).bisoBuyBack() (contracts/core/Kernel.sol#597-598)
        State variables written after the call(s):
        - lastBisoBuyBackTimestamp = block.timestamp (contracts/core/Kernel.sol#598-599)
Reentrancy in Kernel.deploy() (contracts/core/Kernel.sol#563-567):
        External calls:
        - IYieldManager(moduleMap.getModuleAddress(Modules.YieldManager)).deploy() (contracts/core/Kernel.sol#565-566)
        State variables written after the call(s):
        - lastDeployTimestamp = block.timestamp (contracts/core/Kernel.sol#566-567)
Reentrancy in UserPositions.deposit(address,address[],uint256[],uint256) (contracts/core/UserPositions.sol#98-166):
        External calls:
        - erc20.safeTransferFrom(depositor,moduleMap.getModuleAddress(Modules.Kernel),amounts[tokenId]) (contracts/core/UserPositions.sol#130-134)
        - IBlosRewards(moduleMap.getModuleAddress(Modules.BlosRewards)).increaseRewards(tokens[tokenId],depositor,actualAmount) (contracts/core/UserPositions.sol#143-144)
        - IEtherRewards(moduleMap.getModuleAddress(Modules.EtherRewards)).updateUserRewards(tokens[tokenId],depositor) (contracts/core/UserPositions.sol#145-146)
        State variables written after the call(s):
        - _balances[tokens[tokenId]][depositor] += actualAmount (contracts/core/UserPositions.sol#150)
        - _totalSupply[tokens[tokenId]] += actualAmount (contracts/core/UserPositions.sol#149)
Reentrancy in UserPositions.deposit(address,address[],uint256[],uint256) (contracts/core/UserPositions.sol#98-166):
        External calls:
        - IBlosRewards(moduleMap.getModuleAddress(Modules.BlosRewards)).increaseRewards(wethAddress,depositor,ethAmount) (contracts/core/UserPositions.sol#157-158)
        - IEtherRewards(moduleMap.getModuleAddress(Modules.EtherRewards)).updateUserRewards(wethAddress,depositor) (contracts/core/UserPositions.sol#159-160)
        State variables written after the call(s):
        - _balances[wethAddress][depositor] += ethAmount (contracts/core/UserPositions.sol#164)
        - _totalSupply[wethAddress] += ethAmount (contracts/core/UserPositions.sol#163)
Reentrancy in Kernel.distributeEth() (contracts/core/Kernel.sol#588-592):
        External calls:
        - IYieldManager(moduleMap.getModuleAddress(Modules.YieldManager)).distributeEth() (contracts/core/Kernel.sol#588-589)
        State variables written after the call(s):
```

```
getRoleMember(bytes32,uint256) should be declared external:
        - AccessControlUnmarkableUpgradeable.getRoleMember(bytes32,uint256) (node_modules/@openzeppelin/contracts-upgradeable/access/AccessControlUnmarkableUpgradeable.sol#54-56)
getRoleMemberCount(bytes32) should be declared external:
        - AccessControlUnmarkableUpgradeable.getRoleMemberCount(bytes32) (node_modules/@openzeppelin/contracts-upgradeable/access/AccessControlUnmarkableUpgradeable.sol#62-64)
renounceOwnership() should be declared external:
        - OwnableUpgradeable.renounceOwnership() (node_modules/@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol#68-82)
transferOwnership(address) should be declared external:
        - OwnableUpgradeable.transferOwnership(address) (node_modules/@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol#69-73)
initialize(address[],address) should be declared external:
        - BlosRewards.initialize(address[],address) (contracts/core/BlosRewards.sol#29-35)
increaseRewards(address,address,uint256) should be declared external:
        - BlosRewards.increaseRewards(address,address,uint256) (contracts/core/BlosRewards.sol#68-74)
decreaseRewards(address,address,uint256) should be declared external:
        - BlosRewards.decreaseRewards(address,address,uint256) (contracts/core/BlosRewards.sol#76-82)
claimReward(address,address) should be declared external:
        - BlosRewards.claimReward(address,address) (contracts/core/BlosRewards.sol#84-99)
initialize(address[],address) should be declared external:
        - EtherRewards.initialize(address[],address) (contracts/core/EtherRewards.sol#18-24)
initialize(address[],address,address,address) should be declared external:
        - IntegrationMap.initialize(address[],address,address,address) (contracts/core/IntegrationMap.sol#28-40)
isManager(address) should be declared external:
        - Kernel.isManager(address) (contracts/core/Kernel.sol#605-607)
isOwner(address) should be declared external:
        - Kernel.isOwner(address) (contracts/core/Kernel.sol#611-613)
initialize() should be declared external:
        - ModuleMap.initialize() (contracts/core/ModuleMap.sol#11-12)
getModuleAddress(Modules) should be declared external:
        - ModuleMap.getModuleAddress(Modules) (contracts/core/ModuleMap.sol#12-19)
setModuleAddress(Modules,address) should be declared external:
        - ModuleMap.setModuleAddress(Modules,address) (contracts/core/ModuleMap.sol#19-24)
initialize(address[],address) should be declared external:
        - StrategyManager.initialize(address[],address) (contracts/core/StrategyManager.sol#18-23)
initialize(address[],address) should be declared external:
        - StrategyMap.initialize(address[],address) (contracts/core/StrategyMap.sol#45-50)
getUserStrategyBalanceByToken(uint256,address,address) should be declared external:
        - StrategyMap.getUserStrategyBalanceByToken(uint256,address,address) (contracts/core/StrategyMap.sol#384-390)
getTokenTotalBalance(address) should be declared external:
        - StrategyMap.getTokenTotalBalance(address) (contracts/core/StrategyMap.sol#401-408)
initialize(address[],address,address,uint24) should be declared external:
        - SushiSwapTrader.initialize(address[],address,address,address,uint24) (contracts/core/SushiSwapTrader.sol#51-67)
getFactoryAddress() should be declared external:
        - SushiSwapTrader.getFactoryAddress() (contracts/core/SushiSwapTrader.sol#267-269)
getSlippageNumerator() should be declared external:
        - SushiSwapTrader.getSlippageNumerator() (contracts/core/SushiSwapTrader.sol#272-274)
getSlippageDenominator() should be declared external:
        - SushiSwapTrader.getSlippageDenominator() (contracts/core/SushiSwapTrader.sol#277-279)
initialize(address[],address,address,address) should be declared external:
        - UniswapTrader.initialize(address[],address,address,address) (contracts/core/UniswapTrader.sol#71-81)
setPathFor(address,address,uint256,address,uint256) should be declared external:
        - UniswapTrader.setPathFor(address,address,uint256,address,uint256) (contracts/core/UniswapTrader.sol#85-108)
getTokensAndAmountsSorted(address,address,uint256,uint256) should be declared external:
        - UniswapTrader.getTokensAndAmountsSorted(address,address,uint256,uint256) (contracts/core/UniswapTrader.sol#697-723)
initialize(address[],address,uint32) should be declared external:
        - UserPositions.initialize(address[],address,uint32) (contracts/core/UserPositions.sol#39-47)
totalTokenBalance(address) should be declared external:
        - UserPositions.totalTokenBalance(address) (contracts/core/UserPositions.sol#594-601)
getBlosRewardsDuration() should be declared external:
        - UserPositions.getBlosRewardsDuration() (contracts/core/UserPositions.sol#615-617)
initialize(address[],address,uint256,uint32,uint32,uint32,address,address) should be declared external:
```

4.2 AUTOMATED SECURITY SCAN

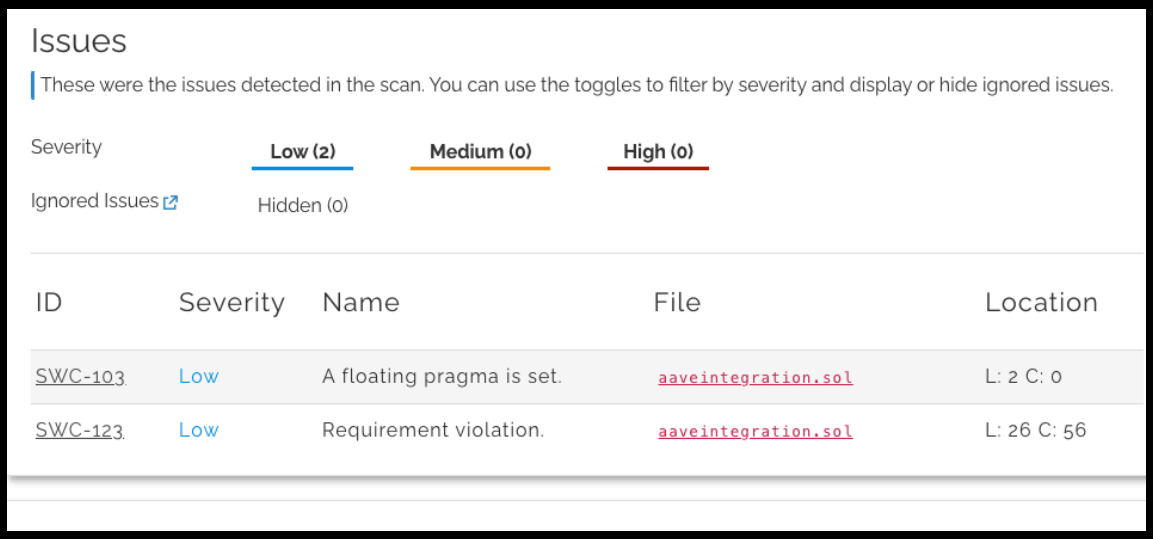
Description:

Halborn used automated security scanners to assist with detection of well-known security issues, and to identify low-hanging fruit on the targets for this engagement. Among the tools used was MythX, a security analysis service for Ethereum smart contracts. MythX performed a scan on the testers machine and sent the compiled results to the analyzers to locate any vulnerabilities. In addition, only security findings are considered as in-scope.

Results:

All the findings were either detected during manual code review or false positive. Only results with findings are shown below.

- AaveIntegration.sol



Issues

These were the issues detected in the scan. You can use the toggles to filter by severity and display or hide ignored issues.

Severity Low (2) Medium (0) High (0)

Ignored Issues Hidden (0)

ID	Severity	Name	File	Location
SWC-103	Low	A floating pragma is set.	aaveintegration.sol	L: 2 C: 0
SWC-123	Low	Requirement violation.	aaveintegration.sol	L: 26 C: 56

- BiosRewards.sol

Issues

These were the issues detected in the scan. You can use the toggles to filter by severity and display or hide ignored issues.

Severity **Low (4)** **Medium (0)** **High (0)**

Ignored Issues Hidden (0)

ID	Severity	Name	File	Location
SWC-103	Low	A floating pragma is set.	biosrewards.sol	L: 2 C: 0
SWC-113	Low	Multiple calls are executed in the same transaction.	biosrewards.sol	L: 119 C: 39
SWC-123	Low	Requirement violation.	biosrewards.sol	L: 139 C: 38
SWC-123	Low	Requirement violation.	biosrewards.sol	L: 17 C: 4

- `DynamicRangeOrdersIntegration.sol`

Issues

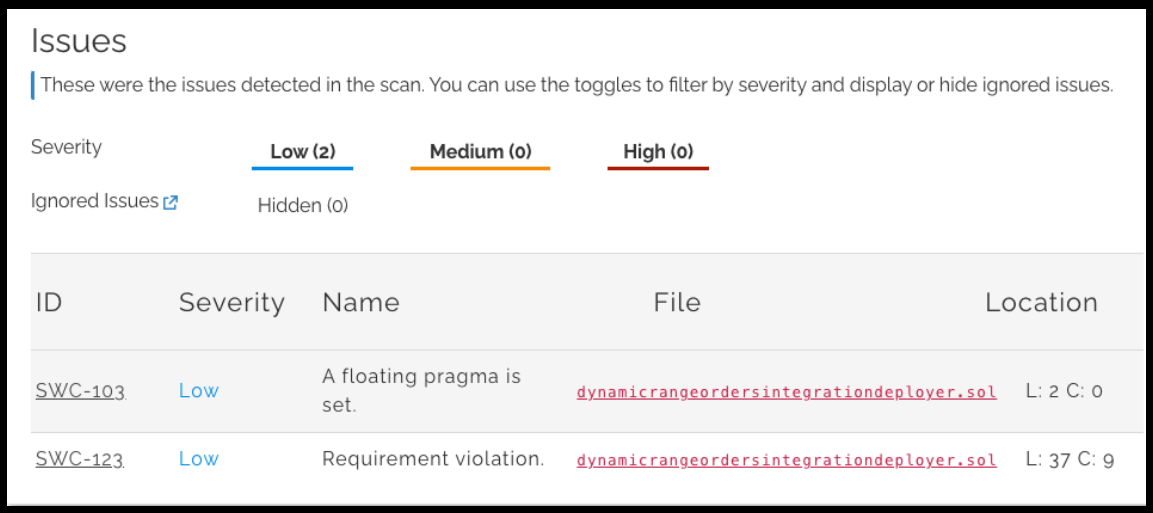
These were the issues detected in the scan. You can use the toggles to filter by severity and display or hide ignored issues.

Severity **Low (6)** **Medium (0)** **High (0)**

Ignored Issues Hidden (0)

ID	Severity	Name	File	Location
SWC-103	Low	A floating pragma is set.	dynamicrangeordersintegration.sol	L: 2 C: 0
SWC-108	Low	State variable visibility is not set.	dynamicrangeordersintegration.sol	L: 47 C: 30
SWC-108	Low	State variable visibility is not set.	dynamicrangeordersintegration.sol	L: 48 C: 30
SWC-108	Low	State variable visibility is not set.	dynamicrangeordersintegration.sol	L: 50 C: 41
SWC-123	Low	Requirement violation.	dynamicrangeordersintegration.sol	L: 622 C: 19
SWC-123	Low	Requirement violation.	dynamicrangeordersintegration.sol	L: 45 C: 18

- DynamicRangeOrdersIntegrationDeployer.sol



Issues

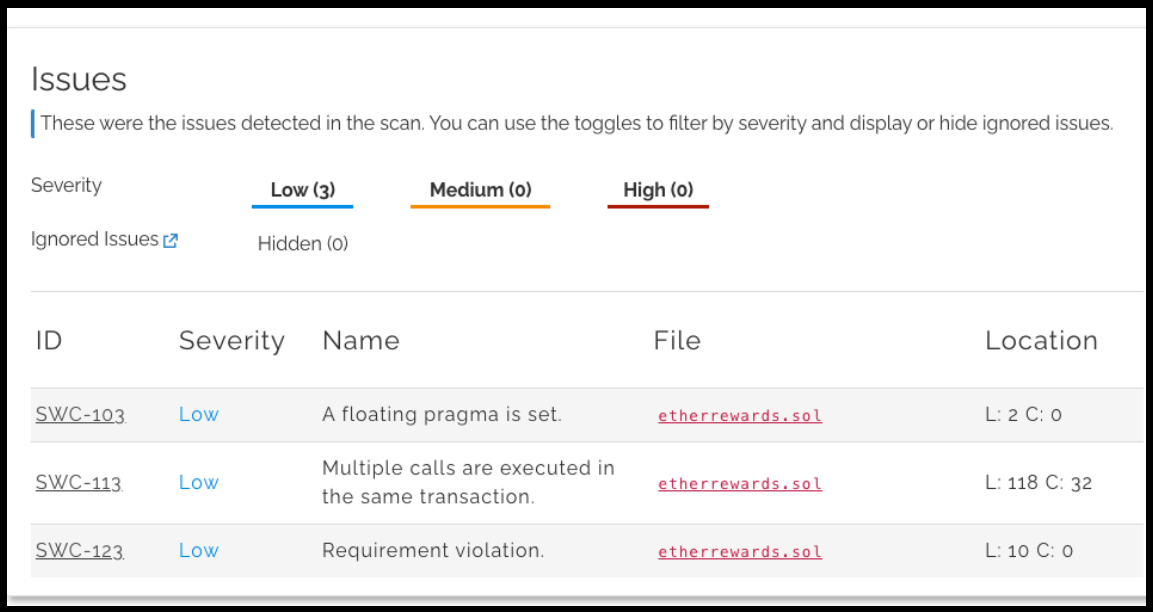
These were the issues detected in the scan. You can use the toggles to filter by severity and display or hide ignored issues.

Severity Low (2) Medium (0) High (0)

Ignored Issues Hidden (0)

ID	Severity	Name	File	Location
SWC-103	Low	A floating pragma is set.	dynamicrangeordersintegrationdeployer.sol	L: 2 C: 0
SWC-123	Low	Requirement violation.	dynamicrangeordersintegrationdeployer.sol	L: 37 C: 9

- EtherRewards.sol



Issues

These were the issues detected in the scan. You can use the toggles to filter by severity and display or hide ignored issues.

Severity Low (3) Medium (0) High (0)

Ignored Issues Hidden (0)

ID	Severity	Name	File	Location
SWC-103	Low	A floating pragma is set.	etherrewards.sol	L: 2 C: 0
SWC-113	Low	Multiple calls are executed in the same transaction.	etherrewards.sol	L: 118 C: 32
SWC-123	Low	Requirement violation.	etherrewards.sol	L: 10 C: 0

- IntegrationMap.sol

Issues

These were the issues detected in the scan. You can use the toggles to filter by severity and display or hide ignored issues.

Severity: **Low (2)** **Medium (0)** **High (0)**

Ignored Issues: Hidden (0)

ID	Severity	Name	File	Location
SWC-103	Low	A floating pragma is set.	integrationmap.sol	L: 2 C: 0
SWC-123	Low	Requirement violation.	integrationmap.sol	L: 18 C: 32

- `Kernel.sol`

Issues

These were the issues detected in the scan. You can use the toggles to filter by severity and display or hide ignored issues.

Severity: **Low (5)** **Medium (0)** **High (0)**

Ignored Issues: Hidden (0)

ID	Severity	Name	File	Location
SWC-103	Low	A floating pragma is set.	kernel.sol	L: 2 C: 0
SWC-107	Low	A call to a user-supplied address is executed.	kernel.sol	L: 549 C: 22
SWC-113	Low	Multiple calls are executed in the same transaction.	kernel.sol	L: 549 C: 22
SWC-123	Low	Requirement violation.	kernel.sol	L: 675 C: 15
SWC-123	Low	Requirement violation.	kernel.sol	L: 34 C: 35

- `ModuleMap.sol`

Issues

These were the issues detected in the scan. You can use the toggles to filter by severity and display or hide ignored issues.

Severity **Low (1)** **Medium (0)** **High (0)**

Ignored Issues Hidden (0)

ID	Severity	Name	File	Location
SWC-103	Low	A floating pragma is set.	modulemap.sol	L: 2 C: 0

- `StrategyManager.sol`

Issues

These were the issues detected in the scan. You can use the toggles to filter by severity and display or hide ignored issues.

Severity **Low (2)** **Medium (0)** **High (0)**

Ignored Issues Hidden (0)

ID	Severity	Name	File	Location
SWC-103	Low	A floating pragma is set.	strategymanager.sol	L: 2 C: 0
SWC-123	Low	Requirement violation.	strategymanager.sol	L: 17 C: 4

- `StrategyMap.sol`

Issues

These were the issues detected in the scan. You can use the toggles to filter by severity and display or hide ignored issues.

Severity **Low (2)** **Medium (0)** **High (0)**

Ignored Issues Hidden (0)

ID	Severity	Name	File	Location
SWC-103	Low	A floating pragma is set.	strategymap.sol	L: 2 C: 0
SWC-123	Low	Requirement violation.	strategymap.sol	L: 18 C: 12

- `SushiSwapTrader.sol`

Issues

These were the issues detected in the scan. You can use the toggles to filter by severity and display or hide ignored issues.

Severity **Low (3)** **Medium (0)** **High (0)**

Ignored Issues Hidden (0)

ID	Severity	Name	File	Location
SWC-103	Low	A floating pragma is set.	sushiswaptrader.sol	L: 2 C: 0
SWC-123	Low	Requirement violation.	sushiswaptrader.sol	L: 108 C: 7
SWC-123	Low	Requirement violation.	sushiswaptrader.sol	L: 27 C: 7

- `UniswapIntegration.sol`

Issues

These were the issues detected in the scan. You can use the toggles to filter by severity and display or hide ignored issues.

Severity **Low (3)** **Medium (0)** **High (0)**

Ignored Issues Hidden (0)

ID	Severity	Name	File	Location
SWC-103	Low	A floating pragma is set.	uniswapintegration.sol	L: 2 C: 0
SWC-108	Low	State variable visibility is not set.	uniswapintegration.sol	L: 45 C: 32
SWC-123	Low	Requirement violation.	uniswapintegration.sol	L: 37 C: 4

- `UniswapIntegrationDeployer.sol`

Issues

These were the issues detected in the scan. You can use the toggles to filter by severity and display or hide ignored issues.

Severity **Low (2)** **Medium (0)** **High (0)**

Ignored Issues Hidden (0)

ID	Severity	Name	File	Location
SWC-103	Low	A floating pragma is set.	uniswaptrader.sol	L: 2 C: 0
SWC-123	Low	Requirement violation.	uniswaptrader.sol	L: 35 C: 12

- `UserPositions.sol`

Issues

These were the issues detected in the scan. You can use the toggles to filter by severity and display or hide ignored issues.

Severity: **Low (3)** **Medium (0)** **High (0)**

Ignored Issues: Hidden (0)

ID	Severity	Name	File	Location
SWC-103	Low	A floating pragma is set.	userpositions.sol	L: 2 C: 0
SWC-123	Low	Requirement violation.	userpositions.sol	L: 550 C: 13
SWC-123	Low	Requirement violation.	userpositions.sol	L: 33 C: 8

- `YearnIntegration.sol`

Issues

These were the issues detected in the scan. You can use the toggles to filter by severity and display or hide ignored issues.

Severity: **Low (4)** **Medium (1)** **High (0)**

Ignored Issues: Hidden (0)

ID	Severity	Name	File	Location
SWC-104	Medium	Unchecked return value from external call.	yearnintegration.sol	L: 194 C: 19
SWC-103	Low	A floating pragma is set.	yearnintegration.sol	L: 2 C: 0
SWC-113	Low	Multiple calls are executed in the same transaction.	yearnintegration.sol	L: 205 C: 17
SWC-123	Low	Requirement violation.	yearnintegration.sol	L: 194 C: 19
SWC-123	Low	Requirement violation.	yearnintegration.sol	L: 26 C: 67

- `YieldManager.sol`

AUTOMATED TESTING

Issues

These were the issues detected in the scan. You can use the toggles to filter by severity and display or hide ignored issues.

Severity **Low (3)** **Medium (0)** **High (0)**

Ignored Issues Hidden (0)

ID	Severity	Name	File	Location
SWC-103	Low	A floating pragma is set.	yieldmanager.sol	L: 2 C: 0
SWC-123	Low	Requirement violation.	yieldmanager.sol	L: 570 C: 30
SWC-123	Low	Requirement violation.	yieldmanager.sol	L: 32 C: 38

THANK YOU FOR CHOOSING

// HALBORN