// HALBORN

0xNodes - SushiSwap Integration - Wave 2

Smart Contract Security Audit

Prepared by: **Halborn**

Date of Engagement: **January 3rd, 2022 - January 10th, 2022**

Visit: **Halborn.com**

DOCUMENT REVISION HISTORY

VERSION	MODIFICATION	DATE	AUTHOR
0.1	Document Creation	01/03/2022	Roberto Reigada
0.2	Document Updates	01/13/2022	Roberto Reigada
0.3	Draft Review	01/13/2022	Gabi Urrutia
1.0	Remediation Plan	01/27/2022	Roberto Reigada
1.1	Remediation Plan Review	01/27/2022	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
Roberto Reigada	Halborn	Roberto.Reigada@halborn.com

EXECUTIVE OVERVIEW

1.1 INTRODUCTION

0xNodes engaged Halborn to conduct a security audit on their smart contracts beginning on January 3rd, 2021 and ending on January 10th. The security assessment was scoped to the smart contract provided in the Github repository 0xNODES/platform/tree/audit.

1.2 AUDIT SUMMARY

The team at Halborn was provided a week for the engagement and assigned a full time security engineer to audit the security of the smart contract. The security 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:

- Ensure that smart contract functions operate as intended
- Identify potential security issues with the smart contracts

In summary, Halborn identified some security risks that were acknowledged by 0xNodes 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 this audit. While manual testing is recommended to uncover flaws in logic, process, and implementation; automated testing techniques help enhance coverage of the bridge 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 assessment of use and safety for the critical Solidity variables and functions in scope to identify any arithmetic related vulnerability classes
- Manual testing by custom scripts
- Scanning of solidity files for vulnerabilities, security hotspots or bugs. (MythX)
- Static Analysis of security for scoped contract, and imported functions. (Slither)
- Testnet deployment (Brownie, 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. The quantitative model ensures repeatable and accurate measurement while enabling users to see the underlying vulnerability characteristics that were 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.

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

1.4 SCOPE

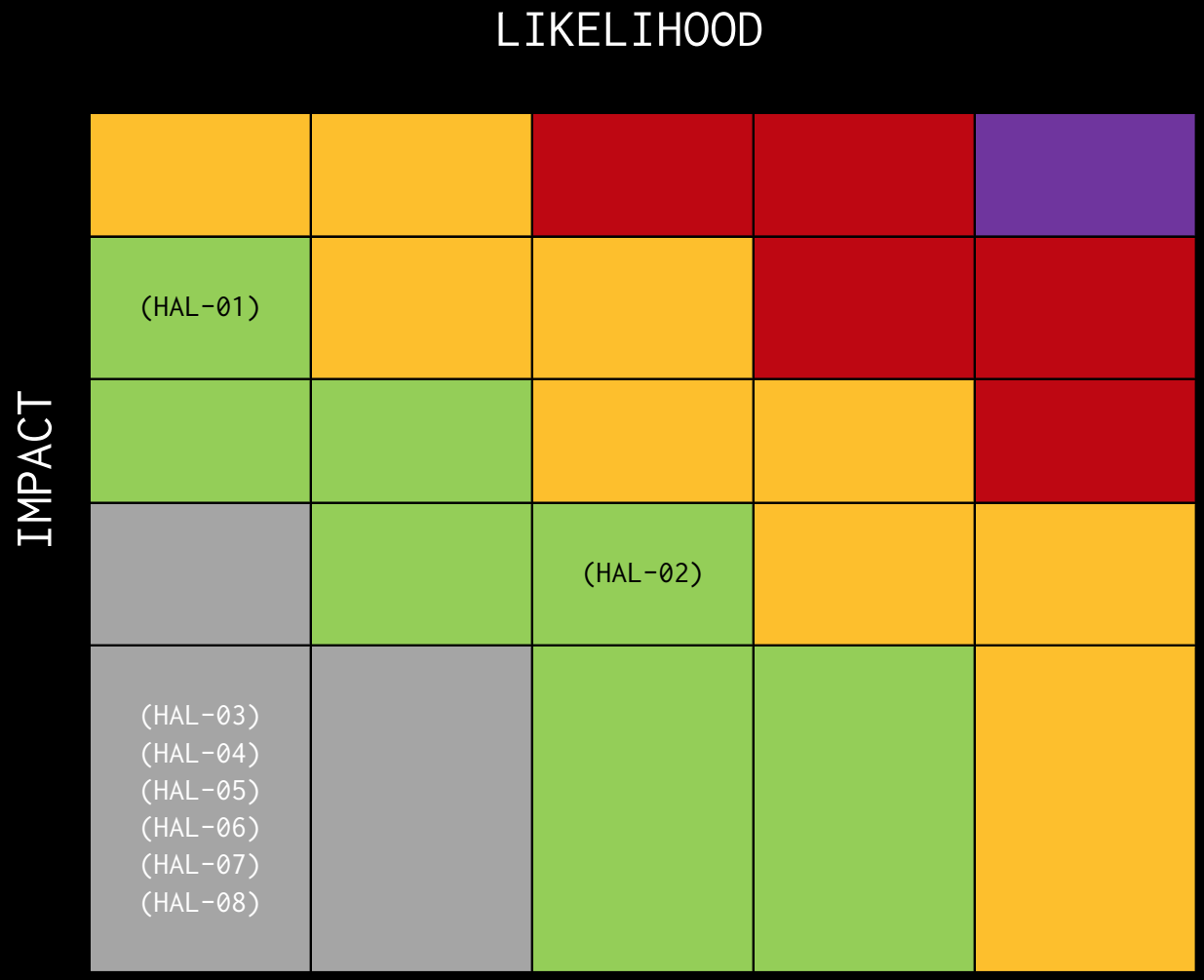
IN-SCOPE:
The security assessment was scoped to the following smart contracts:

- UniswapV3Integration.sol
- Kernel.sol
- IntegrationMap.sol
- Interconnects.sol
- YieldManager.sol

Commit ID: 06a3dd2ddcec58f6a32bb479b4f1b79530eec257

2. ASSESSMENT SUMMARY & FINDINGS OVERVIEW

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL
0	0	0	2	6



SECURITY ANALYSIS	RISK LEVEL	REMEDIATION DATE
(HAL-01) UNCHECKED TRANSFER	Low	RISK ACCEPTED
(HAL-02) MISSING ZERO ADDRESS CHECK	Low	RISK ACCEPTED
(HAL-03) POSSIBLE MISUSE OF PUBLIC FUNCTIONS	Informational	ACKNOWLEDGED
(HAL-04) SOLC 0.8.4 COMPILER VERSION CONTAINS MULTIPLE BUGS	Informational	ACKNOWLEDGED
(HAL-05) CONSTANT KECCAK VARIABLES ARE TREATED AS EXPRESSIONS, NOT CONSTANTS	Informational	ACKNOWLEDGED
(HAL-06) USING ++I CONSUMES LESS GAS THAN I++ IN LOOPS	Informational	ACKNOWLEDGED
(HAL-07) UNNEEDED INITIALIZATION OF UINT256 VARIABLES TO 0	Informational	ACKNOWLEDGED
(HAL-08) UINT32/UINT64 TYPES ARE LESS GAS EFFICIENT	Informational	ACKNOWLEDGED

FINDINGS & TECH DETAILS

3.1 (HAL-01) UNCHECKED TRANSFER - LOW

Description:

In the contract UniswapV3Integration the return value of an external transfer call is not checked. Several tokens do not revert in case of failure and return false. If one of these tokens is used, in case of failure, the withdraw() call would not revert and the Kernel contract would not receive any funds.

Code Location:

UniswapV3Integration.sol

Listing 1: UniswapV3Integration.sol

```
152 // Transfer the funds
153 IERC20MetadataUpgradeable(token).transfer(
154     moduleMap.getModuleAddress(Modules.Kernel),
155     transferAmount // Change to account for partial inavailability
156 );
```

Risk Level:

Likelihood - 1
Impact - 4

Recommendation:

It is recommended to use SafeERC20Upgradeable.

Remediation Plan:

RISK ACCEPTED: The 0xNodes team accepted the risk in this issue.

3.2 (HAL-02) MISSING ZERO ADDRESS CHECK - LOW

Description:

Some initialize functions are missing address validation. Every address should be validated and checked that is different from zero.

Code location:

UniswapV3Integration.sol
- Line 51:
function initialize(address[] memory controllers_, address moduleMap_, address nonfungiblePositionManager_, address uniswapFactory_)

Kernel.sol
- Line 72:
function initialize(address admin_, address owner_, address moduleMap_)
- Line 127:
function addIntegration(address contractAddress, string memory name)
- Line 142:
function addToken(address tokenAddress, bool acceptingDeposits, bool acceptingWithdrawals, bool acceptingLping, bool acceptingBridging, uint256 biosRewardWeight, uint256 reserveRatioNumerator, uint256 targetLiquidityRatioNumerator, uint256 transferFeeKValueNumerator, uint256 transferFeePlatformRatioNumerator)
- Line 314:
function updateTokenRewardWeight(address tokenAddress, uint256 updatedWeight)
- Line 330:
function updateTokenReserveRatioNumerator(address tokenAddress, uint256 reserveRatioNumerator)
- Line 349:
function updateTokenTargetLiquidityRatioNumerator(address tokenAddress, uint256 targetLiquidityRatioNumerator)
- Line 367:

function updateTokenTransferFeeKValueNumerator(address tokenAddress, uint256 transferFeeKValueNumerator)
- Line 386:
function updateTokenTransferFeePlatformRatioNumerator(address tokenAddress, uint256 transferFeePlatformRatioNumerator)
- Line 404:
function updateGasAccount(address payable gasAccount)
- Line 415:
function updateTreasuryAccount(address payable treasuryAccount)
- Line 426:
function updateGasAccountTargetEthBalance(uint256 gasAccountTargetEthBalance)

IntegrationMap.sol
- Line 30:
function initialize(address[] memory controllers_, address moduleMap_, address wethTokenAddress_, address biosTokenAddress_)
- Line 68:
function addIntegration(address contractAddress, string memory name)
- Line 145:
function addToken(address tokenAddress, bool acceptingDeposits, bool acceptingWithdrawals, bool acceptingLping, bool acceptingBridging, uint256 biosRewardWeight, uint256 reserveRatioNumerator, uint256 targetLiquidityRatioNumerator, uint256 transferFeeKValueNumerator, uint256 transferFeePlatformRatioNumerator)

Interconnects.sol
- Line 37:
function initialize(address[] memory controllers_, address moduleMap_, address payable relayAccount_)
- Line 75:
function updateRelayAccount(address payable relayAccount_)

YieldManager.sol
- Line 61:
function initialize(address[] memory controllers_, address moduleMap_, uint256 gasAccountTargetEthBalance_, uint32 biosBuyBackEthWeight_, uint32 treasuryEthWeight_, uint32 protocolFeeEthWeight_, uint32

rewardsEthWeight_, address payable gasAccount_, address payable treasuryAccount_)
- Line 106:
function updateGasAccount(address payable gasAccount_)
- Line 115:
function updateTreasuryAccount(address payable treasuryAccount_)

Risk Level:

Likelihood - 3
Impact - 2

Recommendation:

It is recommended to validate that every address input is different from zero.

Remediation Plan:

RISK ACCEPTED: The 0xNodes team accepted the risk in this issue.

3.3 (HAL-03) POSSIBLE MISUSE OF PUBLIC FUNCTIONS - INFORMATIONAL

Description:

In the following contracts there are functions marked as public but they are never directly called within the same contract or in any of their descendants:

UniswapV3Integration.sol
- initialize() (UniswapV3Integration.sol#51-62)

Kernel.sol
- isManager() (Kernel.sol#644-646)
- isOwner(address) (Kernel.sol#650-652)
- isLiquidityProvider() (Kernel.sol#656-663)

IntegrationMap.sol
- initialize() (IntegrationMap.sol#30-64)

Interconnects.sol
- initialize() (Interconnects.sol#37-44)
- getRelayAccount() (Interconnects.sol#84-86)
- getTokenUserLpBalance() (Interconnects.sol#452-459)
- getTokenPoolLpBalance() (Interconnects.sol#462-469)
- getTokenUserLpFeeRewardBalance() (Interconnects.sol#490-497)
- getTokenLpUsers() (Interconnects.sol#500-507)
- getTokenProtocolFeeRewards() (Interconnects.sol#510-517)

YieldManager.sol
- initialize() (YieldManager.sol#61-80)
- harvestYield() (YieldManager.sol#246-325)
- getReserveTokenBalance() (YieldManager.sol#500-515)
- getDesiredReserveTokenBalance() (YieldManager.sol#519-540)
- getProcessedWethByToken() (YieldManager.sol#577-584)
- getProcessedWethByTokenSum() (YieldManager.sol#587-598)
- getTokenTotalIntegrationBalance() (YieldManager.sol#602-623)

- getGasAccount() (YieldManager.sol#626-628)
- getTreasuryAccount() (YieldManager.sol#631-633)
- getLastEthRewardsAmount() (YieldManager.sol#636-638)
- getGasAccountTargetEthBalance() (YieldManager.sol#641-648)
- getEthDistributionWeights() (YieldManager.sol#654-671)

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

If the functions are not intended to be called internally or by their descendants, it is better to mark all of these functions as external to reduce gas costs.

Remediation Plan:

ACKNOWLEDGED: The 0xNodes team acknowledged this issue.

3.4 (HAL-04) SOLC 0.8.4 COMPILER VERSION CONTAINS MULTIPLE BUGS - INFORMATIONAL

Description:

Solidity compiler version 0.8.9 fixed important bugs in the compiler. The version 0.8.4 is missing this fix:

- 0.8.9

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

It is recommended to use the most tested and stable versions, such as 0.6.12 or 0.7.6. Otherwise, if you still want to use ^0.8.0, because of the new functionality it provides, it is recommended to use ^0.8.9 version.

Remediation Plan:

ACKNOWLEDGED: The 0xNodes team acknowledged this issue.

3.5 (HAL-05) CONSTANT KECCAK VARIABLES ARE TREATED AS EXPRESSIONS, NOT CONSTANTS - INFORMATIONAL

Description:

In the contract Kernel, the roles are declared the following way:

Listing 2: Kernel.sol

```
33 bytes32 public constant OWNER_ROLE = keccak256("owner_role");
34 bytes32 public constant MANAGER_ROLE = keccak256("manager_role");
```

Listing 3: Kernel.sol

```
45     bytes32 public constant LIQUIDITY_PROVIDER_ROLE =
46         keccak256("liquidity_provider_role");
```

This results in the keccak256 operation being performed whenever the variable is used, increasing gas costs relative to just storing the output hash.

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

It is recommended to either:
1. Keep the variables as constant and hard-code the bytes32 string into the smart contracts.
2. Declare all the roles as immutable and perform the hashing assignment

in the constructors.

Remediation Plan:

ACKNOWLEDGED: The 0xNodes team acknowledged this issue.

3.6 (HAL-06) USING ++I CONSUMES LESS GAS THAN I++ IN LOOPS - INFORMATIONAL

Description:

In all the loops, the variable `i` is incremented using `i++`. It is known that, in loops, using `++i` costs less gas per iteration than `i++`.

Code Location:

UniswapV3Integration.sol
- Line 298: `for (uint32 i = 1; i <= poolIDCounter; i++)`

IntegrationMap.sol
- Line 540: `for (uint256 tokenId; tokenId < tokenAddresses.length; tokenId++){`

Interconnects.sol
- Line 65: `for (uint256 i = 0; i < tokens.length; i++){`
- Line 111: `for (uint256 tokenId; tokenId < tokens.length; tokenId++){`
- Line 164: `for (uint256 tokenId; tokenId < tokens.length; tokenId++){`
- Line 190: `lpUserIdx++`
- Line 226: `for (uint256 tokenId; tokenId < tokens.length; tokenId++){`
- Line 283: `for (uint256 i; i < tokens.length; i++){`
- Line 311: `for (uint256 i; i < tokens.length; i++){`
- Line 329: `for (uint256 i; i < tokens.length; i++){`
- Line 356: `for (uint256 tokenId; tokenId < tokens.length; tokenId++){`
- Line 388: `lpUserIdx++`
- Line 425: `for (uint256 tokenId; tokenId < tokens.length; tokenId++){`
- Line 481:
`for (uint256 lpUserIdx; lpUserIdx < lpUsers.length; lpUserIdx++){`

YieldManager.sol
- Line 133: `for (uint256 i = 0; i < deployments.length; i++){`
- Line 138: `for (uint256 j = 0; j < deployments[i].tokens.length; j++){`

- Line 272: tokenIterator++
- Line 440: for (uint256 tokenId; tokenId < tokenCount; tokenId++){
- Line 567: for (uint256 tokenId; tokenId < tokenCount; tokenId++){
- Line 593:
for (uint256 tokenId; tokenId < tokenAddresses.length; tokenId++){
- Line 617: integrationId++
- Line 706: integrationId++

Proof of Concept:

For example, based in the following test contract:

Listing 4: Test.sol

```
1 //SPDX-License-Identifier: MIT
2 pragma solidity 0.8.9;
3
4 contract test {
5     function postiincrement(uint256 iterations) public {
6         for (uint256 i = 0; i < iterations; i++) {
7         }
8     }
9     function preiincrement(uint256 iterations) public {
10        for (uint256 i = 0; i < iterations; ++i) {
11        }
12    }
13 }
```

We can see the difference in the gas costs:

```
>>> test_contract.postiincrement(1)
Transaction sent: 0xlecede6b109b707786d3685bd71dd9f22dc389957653036ca04c4cd2e72c5e0b
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 44
  test.postiincrement confirmed   Block: 13622335   Gas used: 21620 (0.32%)

<Transaction '0xlecede6b109b707786d3685bd71dd9f22dc389957653036ca04c4cd2e72c5e0b'>
>>> test_contract.preiincrement(1)
Transaction sent: 0x205f09a4d2268de4c1a40f35bb2ec2847bf2ab8d584909b42c71a022b047614a
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 45
  test.preiincrement confirmed   Block: 13622336   Gas used: 21593 (0.32%)

<Transaction '0x205f09a4d2268de4c1a40f35bb2ec2847bf2ab8d584909b42c71a022b047614a'>
>>> test_contract.postiincrement(10)
Transaction sent: 0x98c04430526a59balcf947c114b62666a4417165947d31bf300cd6ae68328033
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 46
  test.postiincrement confirmed   Block: 13622337   Gas used: 22673 (0.34%)

<Transaction '0x98c04430526a59balcf947c114b62666a4417165947d31bf300cd6ae68328033'>
>>> test_contract.preiincrement(10)
Transaction sent: 0xf060d04714eff8482a828342414d5a20be9958c822d42860e7992aba20elde05
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 47
  test.preiincrement confirmed   Block: 13622338   Gas used: 22601 (0.34%)

<Transaction '0xf060d04714eff8482a828342414d5a20be9958c822d42860e7992aba20elde05'>
```

Risk Level:

Likelihood - 1

Impact - 1

Recommendation:

It is recommended to use ++i instead of i++ to increment the value of an uint variable inside a loop. This is not applicable outside of loops.

Remediation Plan:

ACKNOWLEDGED: The 0xNodes team acknowledged this issue.

3.7 (HAL-07) UNNEEDED INITIALIZATION OF UINT256 VARIABLES TO 0 - INFORMATIONAL

Description:

`uint256` variables are already initialized to `0` by default. `uint256 i = 0`, for example, reassigns the `0` to `i` which wastes gas.

Code Location:

UniswapV3Integration.sol
- Line 107: `uint256 amount0Withdrawn = 0;`
- Line 108: `uint256 amount1Withdrawn = 0;`
- Line 247: `uint256 amountA = 0;`
- Line 248: `uint256 amountB = 0;`

Interconnects.sol
- Line 65: `for (uint256 i = 0; i < tokens.length; i++){`
- Line 479: `uint256 sum = 0;`

YieldManager.sol
- Line 133: `for (uint256 i = 0; i < deployments.length; i++){`
- Line 138: `for (uint256 j = 0; j < deployments[i].tokens.length; j++){`

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

It is recommended to not initialize `uint256` variables to `0` in order to save some gas. For example: `for (uint256 i; i < tokens.length; ++i){`

Remediation Plan:

ACKNOWLEDGED: The 0xNodes team acknowledged this issue.

3.8 (HAL-08) UINT32/UINT64 TYPES ARE LESS GAS EFFICIENT - INFORMATIONAL

Description:

uint32, uint64. . . variables are less gas efficient than uint256. Due to how the EVM natively works on 256-bit numbers, using for example 32-bit number introduces additional costs as the EVM has to properly enforce the limits of this smaller type.

This happens in multiple functions across the different smart contracts:

UniswapV3Integration.sol
- Line 37: uint32 public override poolIDCounter;
- Line 40: mapping(uint32 => PositionNFT)internal pools;
- Line 43:
mapping(uint32 => mapping(address => uint256))public override balances;
- Line 64:
function _verifyPoolAndTokens(address token, uint32 poolID)internal
view {
- Line 72: modifier verifyPoolAndTokens(address token, uint32 poolID){
- Line 81: uint32 poolID
- Line 99: uint32 poolID
- Line 234: function deploy(uint32 poolID)external override {
- Line 239: uint32 poolID,
- Line 298: for (uint32 i = 1; i <= poolIDCounter; i++){
- Line 317: function getPoolBalance(uint32 poolID)
- Line 337: function getPool(uint32 poolID)
- Line 391: function getPendingYield(uint32 poolID)

Kernel.sol
- Line 103: function setBiosRewardsDuration(uint32 biosRewardsDuration)
- Line 223: uint32 biosBuyBackEthWeight,
- Line 224: uint32 treasuryEthWeight,
- Line 225: uint32 protocolFeeEthWeight,
- Line 226: uint32 rewardsEthWeight

IntegrationMap.sol
- Line 15:
uint32 private constant RESERVE_RATIO_DENOMINATOR = 1_000_000;
- Line 26:
uint32 private constant TARGET_LIQUIDITY_RATIO_DENOMINATOR = 1_000_000;
- Line 27:
uint32 private constant TRANSFER_FEE_K_VALUE_DENOMINATOR = 1_000_000;
- Line 28:
uint32 private constant TRANSFER_FEE_PLATFORM_RATIO_DENOMINATOR = 1
_000_000;
- Line 669: returns (uint32)
- Line 694: returns (uint32)
- Line 719: returns (uint32)
- Line 744: returns (uint32)

Interconnects.sol
- Line 416: uint32 targetLiquidityRatioDenominator = integrationMap
- Line 418: uint32 kValueDenominator = integrationMap
- Line 420: uint32 protocolFeeDenominator = integrationMap

YieldManager.sol
- Line 32: uint32 private biosBuyBackEthWeight;
- Line 33: uint32 private treasuryEthWeight;
- Line 34: uint32 private protocolFeeEthWeight;
- Line 35: uint32 private rewardsEthWeight;
- Line 65: uint32 biosBuyBackEthWeight_,
- Line 66: uint32 treasuryEthWeight_,
- Line 67: uint32 protocolFeeEthWeight_,
- Line 68: uint32 rewardsEthWeight_,
- Line 94: uint32 biosBuyBackEthWeight_,
- Line 95: uint32 treasuryEthWeight_,
- Line 96: uint32 protocolFeeEthWeight_,
- Line 97: uint32 rewardsEthWeight_
- Line 547: returns (uint32 ethWeightSum)
- Line 659: uint32,
- Line 660: uint32,
- Line 661: uint32,
- Line 662: uint32,

In general, the usage of these smaller types only improves the gas costs in cases where the variables can be packed together, for example structs.

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

It is recommended to make use of uint256 variables, for example, in state variables that are not part/packed in any struct to reduce gas costs.

Remediation Plan:

ACKNOWLEDGED: The 0xNodes team acknowledged this issue.

MANUAL TESTING

4.1 CONTRACT INITIALIZATION

Contract	Inherits from	Has initialize function?	Has initializer modifier?	Code
Kernel	Initializable, AccessControlEnumerableUpgradeable, ModuleMapConsumer, IKernel, ReentrancyGuardUpgradeable	Yes. The initialize function correctly initializes ModuleMapConsumer, ReentrancyGuardUpgradeable and AccessControlEnumerableUpgradeable	Yes	
UniswapV3Integration	Initializable, ModuleMapConsumer, Controlled, IAMMIntegration, IUniswapV3Integration, IERC721Receiver	Yes. The initialize function correctly initializes Controlled, which itself initializes also ModuleMapConsumer	Yes	
IntegrationMap	Initializable, ModuleMapConsumer, Controlled, IIntegrationMap	Yes. The initialize function correctly initializes Controlled, which itself initializes also ModuleMapConsumer	Yes	
Interconnects	Initializable, ModuleMapConsumer, Controlled, IInterconnects, ReentrancyGuardUpgradeable	Yes. The initialize function correctly initializes Controlled, which itself initializes also ModuleMapConsumer, although ReentrancyGuardUpgradeable is not initialized. Even if leaving this contract uninitialized has no security impact it is still recommended and it is considered a security best practice to initialize it.	Yes	
YieldManager	Initializable, ModuleMapConsumer, Controlled, IYieldManager	Yes. The initialize function correctly initializes Controlled, which itself initializes also ModuleMapConsumer	Yes	

```
72      function initialize(
73          address admin_,
74          address owner_,
75          address moduleMap_
76      ) external initializer {
77          _ModuleMapConsumer_init(moduleMap_);
78          __ReentrancyGuard_init();
79          __AccessControl_init();
80          // make the "admin " address the default admin role
81          _setupRole(DEFAULT_ADMIN_ROLE, admin_);
82
83          // make the "owner " address the owner of the system
84          _setupRole(OWNER_ROLE, owner_);
85
86          // give the "owner " address the manager role, too
87          _setupRole(MANAGER_ROLE, owner_);
88
89          // give the "owner " address the liquidity provider role, too
90          _setupRole(LIQUIDITY_PROVIDER_ROLE, owner_);
91
92          // owners are admins of managers
93          _setRoleAdmin(MANAGER_ROLE, OWNER_ROLE);
94
95          // managers are admins of liquidity providers
96          _setRoleAdmin(LIQUIDITY_PROVIDER_ROLE, MANAGER_ROLE);
97
98          initializationTimestamp = block.timestamp;
99          lpWhitelistEnabled = true;
100     }
```

```
51      function initialize(
52          address[] memory controllers_,
53          address moduleMap_,
54          address nonfungiblePositionManager_,
55          address uniswapFactory_
56      ) public initializer {
57          __Controlled_init(controllers_, moduleMap_);
58          positionManager = INonfungiblePositionManager(
59              nonfungiblePositionManager_
60          );
61          factory = IUniswapV3Factory(uniswapFactory_);
62      }
```

```
30      function initialize(
31          address[] memory controllers_,
32          address moduleMap_,
33          address wethTokenAddress_,
34          address biosTokenAddress_
35      ) public initializer {
36          __Controlled_init(controllers_, moduleMap_);
37          wethTokenAddress = wethTokenAddress_;
38          biosTokenAddress = biosTokenAddress_;
39
40          _addToken(
41              wethTokenAddress_,
42              true,
43              true,
44              true,
45              true,
46              1_000,
47              50_000,
48              100_000,
49              40_000,
50              50_000
51          );
52          _addToken(
53              biosTokenAddress_,
54              true,
55              true,
56              true,
57              true,
58              1_000,
59              0,
60              100_000,
61              40_000,
62              50_00
63          );
64      }
```

```
37      function initialize(
38          address[] memory controllers_,
39          address moduleMap_,
40          address payable relayAccount_
41      ) public initializer {
42          __Controlled_init(controllers_, moduleMap_);
43          relayAccount = relayAccount_;
44      }
```

```
61      function initialize(
62          address[] memory controllers_,
63          address moduleMap_,
64          uint256 gasAccountTargetEthBalance_,
65          uint32 biosBuyBackEthWeight_,
66          uint32 treasuryEthWeight_,
67          uint32 protocolFeeEthWeight_,
68          uint32 rewardsEthWeight_,
69          address payable gasAccount_,
70          address payable treasuryAccount_
71      ) public initializer {
72          __Controlled_init(controllers_, moduleMap_);
73          gasAccountTargetEthBalance = gasAccountTargetEthBalance_;
74          biosBuyBackEthWeight = biosBuyBackEthWeight_;
75          treasuryEthWeight = treasuryEthWeight_;
76          protocolFeeEthWeight = protocolFeeEthWeight_;
77          rewardsEthWeight = rewardsEthWeight_;
78          gasAccount = gasAccount_;
79          treasuryAccount = treasuryAccount_;
80      }
```

No major issues found in the initialization, although, it is recommended to initialize `ReentrancyGuardUpgradeable` in the `Interconnects` contract by calling `__ReentrancyGuard_init()`.

AUTOMATED TESTING

5.1 STATIC ANALYSIS REPORT

Description:

Halborn used automated testing techniques to enhance the coverage of certain areas of the scoped contracts. 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, Slither was run on the all-scoped contracts. 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.

Slither results:

Slither only found some issues in the following smart contracts:

Kernel.sol

```
Kernel.withdraw(address[],uint256[],bool) (contracts/core/Kernel.sol#459-479) sends eth to arbitrary user
	Dangerous calls:
	- address(msg.sender).transfer(ethWithdrawn) (contracts/core/Kernel.sol#475)
Kernel.withdrawAllAndClaim(address[],bool) (contracts/core/Kernel.sol#488-525) sends eth to arbitrary user
	Dangerous calls:
	- address(msg.sender).transfer(ethWithdrawn + ethClaimed) (contracts/core/Kernel.sol#513)
Kernel.claimEthRewards() (contracts/core/Kernel.sol#562-574) sends eth to arbitrary user
	Dangerous calls:
	- address(msg.sender).transfer(ethClaimed) (contracts/core/Kernel.sol#571)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#functions-that-send-ether-to-arbitrary-destinations

Reentrancy in Kernel.claimAllRewards() (contracts/core/Kernel.sol#593-600):
	External calls:
	- ethClaimed = claimEthRewards() (contracts/core/Kernel.sol#598)
		- ethClaimed = IUserPositions(moduleMap.getModuleAddress(Modules.UserPositions)).claimEthRewards(msg.sender) (contracts/core/Kernel.sol#567-569)
	- biosClaimed = claimBiosRewards() (contracts/core/Kernel.sol#599)
		- biosClaimed = IBiosRewards(moduleMap.getModuleAddress(Modules.BiosRewards)).claimBiosRewards(msg.sender) (contracts/core/Kernel.sol#583-585)
	External calls sending eth:
	- ethClaimed = claimEthRewards() (contracts/core/Kernel.sol#598)
		- address(msg.sender).transfer(ethClaimed) (contracts/core/Kernel.sol#571)
	State variables written after the call(s):
	- biosClaimed = claimBiosRewards() (contracts/core/Kernel.sol#599)
		- _status = _ENTERED (node_modules/@openzeppelin/contracts-upgradeable/security/ReentrancyGuardUpgradeable.sol#59)
		- _status = _NOT_ENTERED (node_modules/@openzeppelin/contracts-upgradeable/security/ReentrancyGuardUpgradeable.sol#65)
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#35)
	- ContextUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#31)
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#31)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variable-shadowing

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#71)
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#95)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return

Reentrancy in Kernel.biosBuyBack() (contracts/core/Kernel.sol#635-640):
	External calls:
	- IYieldManager(moduleMap.getModuleAddress(Modules.YieldManager)).biosBuyBack() (contracts/core/Kernel.sol#636-637)
	State variables written after the call(s):
	- lastBiosBuyBackTimestamp = block.timestamp (contracts/core/Kernel.sol#638)
Reentrancy in Kernel.deploy(IYieldManager.DeployRequest[]) (contracts/core/Kernel.sol#603-612):
	External calls:
	- IYieldManager(moduleMap.getModuleAddress(Modules.YieldManager)).deploy(deployments) (contracts/core/Kernel.sol#607-609)
	State variables written after the call(s):
	- lastDeployTimestamp = block.timestamp (contracts/core/Kernel.sol#610)
Reentrancy in Kernel.distributeEth() (contracts/core/Kernel.sol#626-632):
	External calls:
	- IYieldManager(moduleMap.getModuleAddress(Modules.YieldManager)).distributeEth() (contracts/core/Kernel.sol#627-628)
	State variables written after the call(s):
	- lastDistributeEthTimestamp = block.timestamp (contracts/core/Kernel.sol#630)
	- lastLastDistributeEthTimestamp = lastDistributeEthTimestamp (contracts/core/Kernel.sol#629)
Reentrancy in Kernel.harvestYield(address,address[]) (contracts/core/Kernel.sol#615-623):
	External calls:

	- IYieldManager(moduleMap.getModuleAddress(Modules.YieldManager)).harvestYield(integrationAddress,tokenAddresses) (contracts/core/Kernel.sol#619-620)
	State variables written after the call(s):
	- lastHarvestYieldTimestamp = block.timestamp (contracts/core/Kernel.sol#621)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2

Reentrancy in Kernel.addIntegration(address,string) (contracts/core/Kernel.sol#127-135):
	External calls:
	- IIntegrationMap(moduleMap.getModuleAddress(Modules.IntegrationMap)).addIntegration(contractAddress,name) (contracts/core/Kernel.sol#131-132)
	Event emitted after the call(s):
	- IntegrationAdded(contractAddress,name) (contracts/core/Kernel.sol#134)
Reentrancy in Kernel.addToken(address,bool,bool,bool,bool,uint256,uint256,uint256,uint256) (contracts/core/Kernel.sol#142-216):
	External calls:
	- IIntegrationMap(moduleMap.getModuleAddress(Modules.IntegrationMap)).addToken(tokenAddress,acceptingDeposits,acceptingWithdrawals,acceptingIping,acceptingBridging,biosRewardWeight,reserveRatioNumerator,targetLiquidityRatioNumerator,transferFeeKValueNumerator,transferFeePlatformRatioNumerator) (contracts/core/Kernel.sol#154-166)
	- IERC20MetadataUpgradeable(tokenAddress).safeApprove(moduleMap.getModuleAddress(Modules.YieldManager),type()(uint256).max) (contracts/core/Kernel.sol#174-177)
	- IERC20MetadataUpgradeable(tokenAddress).safeApprove(moduleMap.getModuleAddress(Modules.UserPositions),type()(uint256).max) (contracts/core/Kernel.sol#186-189)
	- IERC20MetadataUpgradeable(tokenAddress).safeApprove(moduleMap.getModuleAddress(Modules.SwapManager),type()(uint256).max) (contracts/core/Kernel.sol#198-201)
	Event emitted after the call(s):
	- TokenAdded(tokenAddress,acceptingDeposits,acceptingWithdrawals,acceptingIping,acceptingBridging,biosRewardWeight,reserveRatioNumerator,targetLiquidityRatioNumerator,transferFeeKValueNumerator,transferFeePlatformRatioNumerator)
(contracts/core/Kernel.sol#204-215)
Reentrancy in Kernel.biosBuyBack() (contracts/core/Kernel.sol#635-640):
	External calls:
	- IYieldManager(moduleMap.getModuleAddress(Modules.YieldManager)).biosBuyBack() (contracts/core/Kernel.sol#636-637)
	Event emitted after the call(s):
	- BiosBuyBack() (contracts/core/Kernel.sol#639)
```

```
Reentrancy in Kernel.claimAllRewards() (contracts/core/Kernel.sol#593-600):
        External calls:
        - ethClaimed = claimEthRewards() (contracts/core/Kernel.sol#598)
            - ethClaimed = IUserPositions(moduleMap.getModuleAddress(Modules.UserPositions)).claimEthRewards(msg.sender) (contracts/core/Kernel.sol#567-569)
        - biosClaimed = claimBiosRewards() (contracts/core/Kernel.sol#599)
            - biosClaimed = IBiosRewards(moduleMap.getModuleAddress(Modules.BiosRewards)).claimBiosRewards(msg.sender) (contracts/core/Kernel.sol#583-585)
        External calls sending eth:
        - ethClaimed = claimEthRewards() (contracts/core/Kernel.sol#598)
            - address(msg.sender).transfer(ethClaimed) (contracts/core/Kernel.sol#571)
        Event emitted after the call(s):
        - ClaimBiosRewards(msg.sender,biosClaimed) (contracts/core/Kernel.sol#587)
            - biosClaimed = claimBiosRewards() (contracts/core/Kernel.sol#599)
Reentrancy in Kernel.claimBiosRewards() (contracts/core/Kernel.sol#578-588):
        External calls:
        - biosClaimed = IBiosRewards(moduleMap.getModuleAddress(Modules.BiosRewards)).claimBiosRewards(msg.sender) (contracts/core/Kernel.sol#583-585)
        Event emitted after the call(s):
        - ClaimBiosRewards(msg.sender,biosClaimed) (contracts/core/Kernel.sol#587)
Reentrancy in Kernel.claimEthRewards() (contracts/core/Kernel.sol#562-574):
        External calls:
        - ethClaimed = IUserPositions(moduleMap.getModuleAddress(Modules.UserPositions)).claimEthRewards(msg.sender) (contracts/core/Kernel.sol#567-569)
        External calls sending eth:
        - address(msg.sender).transfer(ethClaimed) (contracts/core/Kernel.sol#571)
        Event emitted after the call(s):
        - ClaimEthRewards(msg.sender,ethClaimed) (contracts/core/Kernel.sol#573)
Reentrancy in Kernel.deploy(ITYieldManager.DeployRequest[]) (contracts/core/Kernel.sol#603-612):
        External calls:
        - IYieldManager(moduleMap.getModuleAddress(Modules.YieldManager)).deploy(deployments) (contracts/core/Kernel.sol#607-609)
        Event emitted after the call(s):
        - Deploy() (contracts/core/Kernel.sol#611)
Reentrancy in Kernel.distributeEth() (contracts/core/Kernel.sol#626-632):
        External calls:
        - IYieldManager(moduleMap.getModuleAddress(Modules.YieldManager)).distributeEth() (contracts/core/Kernel.sol#627-628)
        Event emitted after the call(s):
        - DistributeEth() (contracts/core/Kernel.sol#631)
Reentrancy in Kernel.harvestYield(address,address[]) (contracts/core/Kernel.sol#615-623):
        External calls:
        - IYieldManager(moduleMap.getModuleAddress(Modules.YieldManager)).harvestYield(integrationAddress,tokenAddresses) (contracts/core/Kernel.sol#619-620)
        Event emitted after the call(s):
        - HarvestYield() (contracts/core/Kernel.sol#622)
Reentrancy in Kernel.seedBiosRewards(uint256) (contracts/core/Kernel.sol#114-122):
        External calls:
        - IBiosRewards(moduleMap.getModuleAddress(Modules.BiosRewards)).seedBiosRewards(msg.sender,biosAmount) (contracts/core/Kernel.sol#118-119)
        Event emitted after the call(s):
        - SeedBiosRewards(biosAmount) (contracts/core/Kernel.sol#121)
Reentrancy in Kernel.setBiosRewardsDuration(uint32) (contracts/core/Kernel.sol#103-111):
        External calls:
        - IBiosRewards(moduleMap.getModuleAddress(Modules.BiosRewards)).setBiosRewardsDuration(biosRewardsDuration) (contracts/core/Kernel.sol#107-108)
        Event emitted after the call(s):
        - SetBiosRewardsDuration(biosRewardsDuration) (contracts/core/Kernel.sol#110)
Reentrancy in Kernel.updateEthDistributionWeights(uint32,uint32,uint32,uint32) (contracts/core/Kernel.sol#222-242):
        External calls:
        - IYieldManager(moduleMap.getModuleAddress(Modules.YieldManager)).updateEthDistributionWeights(biosBuyBackEthWeight,treasuryEthWeight,protocolFeeEthWeight,rewardsEthWeight) (contracts/core/Kernel.sol#228-234)
        Event emitted after the call(s):
        - EthDistributionWeightsUpdated(biosBuyBackEthWeight,treasuryEthWeight,protocolFeeEthWeight,rewardsEthWeight) (contracts/core/Kernel.sol#236-241)
Reentrancy in Kernel.updateGasAccount(address) (contracts/core/Kernel.sol#404-412):
        External calls:
        - IYieldManager(moduleMap.getModuleAddress(Modules.YieldManager)).updateGasAccount(gasAccount) (contracts/core/Kernel.sol#408-409)
        Event emitted after the call(s):
        - GasAccountUpdated(gasAccount) (contracts/core/Kernel.sol#411)
Reentrancy in Kernel.updateGasAccountTargetEthBalance(uint256) (contracts/core/Kernel.sol#426-433):
        External calls:
        - IYieldManager(moduleMap.getModuleAddress(Modules.YieldManager)).updateGasAccountTargetEthBalance(gasAccountTargetEthBalance) (contracts/core/Kernel.sol#429-430)
        Event emitted after the call(s):
        - GasAccountTargetEthBalanceUpdated(gasAccountTargetEthBalance) (contracts/core/Kernel.sol#432)
Reentrancy in Kernel.updateTokenReserveRatioNumerator(address,uint256) (contracts/core/Kernel.sol#330-345):
        External calls:
        - ITIntegrationMap(moduleMap.getModuleAddress(Modules.IntegrationMap)).updateTokenReserveRatioNumerator(tokenAddress,reserveRatioNumerator) (contracts/core/Kernel.sol#334-338)
        Event emitted after the call(s):
        - TokenSettingUpdated(tokenAddress,TokenSettings.TokenSettingName.reserveRatioNumerator,reserveRatioNumerator) (contracts/core/Kernel.sol#340-344)
Reentrancy in Kernel.updateTokenRewardWeight(address,uint256) (contracts/core/Kernel.sol#314-326):
        External calls:
        - ITIntegrationMap(moduleMap.getModuleAddress(Modules.IntegrationMap)).updateTokenRewardWeight(tokenAddress,updatedWeight) (contracts/core/Kernel.sol#318-319)
        Event emitted after the call(s):
        - TokenSettingUpdated(tokenAddress,TokenSettings.TokenSettingName.rewardWeight,updatedWeight) (contracts/core/Kernel.sol#321-325)
Reentrancy in Kernel.updateTokenTargetLiquidityRatioNumerator(address,uint256) (contracts/core/Kernel.sol#349-363):
        External calls:
        - ITIntegrationMap(moduleMap.getModuleAddress(Modules.IntegrationMap)).updateTokenTargetLiquidityRatioNumerator(tokenAddress,targetLiquidityRatioNumerator) (contracts/core/Kernel.sol#353-357)
        Event emitted after the call(s):
        - TokenSettingUpdated(tokenAddress,TokenSettings.TokenSettingName.targetLiquidityRatioNumerator,targetLiquidityRatioNumerator) (contracts/core/Kernel.sol#358-362)
Reentrancy in Kernel.updateTokenTransferFeeKValueNumerator(address,uint256) (contracts/core/Kernel.sol#367-382):
        External calls:
        - ITIntegrationMap(moduleMap.getModuleAddress(Modules.IntegrationMap)).updateTokenTransferFeeKValueNumerator(tokenAddress,transferFeeKValueNumerator) (contracts/core/Kernel.sol#371-375)
        Event emitted after the call(s):
        - TokenSettingUpdated(tokenAddress,TokenSettings.TokenSettingName.transferFeeKValueNumerator,transferFeeKValueNumerator) (contracts/core/Kernel.sol#377-381)
Reentrancy in Kernel.updateTokenTransferFeePlatformRatioNumerator(address,uint256) (contracts/core/Kernel.sol#386-401):
        External calls:
        - ITIntegrationMap(moduleMap.getModuleAddress(Modules.IntegrationMap)).updateTokenTransferFeePlatformRatioNumerator(tokenAddress,transferFeePlatformRatioNumerator) (contracts/core/Kernel.sol#390-394)
        Event emitted after the call(s):
        - TokenSettingUpdated(tokenAddress,TokenSettings.TokenSettingName.transferFeePlatformRatioNumerator,transferFeePlatformRatioNumerator) (contracts/core/Kernel.sol#396-400)
Reentrancy in Kernel.updateTreasuryAccount(address) (contracts/core/Kernel.sol#415-423):
        External calls:
        - IYieldManager(moduleMap.getModuleAddress(Modules.YieldManager)).updateTreasuryAccount(treasuryAccount) (contracts/core/Kernel.sol#419-420)
        Event emitted after the call(s):
        - TreasuryAccountUpdated(treasuryAccount) (contracts/core/Kernel.sol#422)
Reentrancy in Kernel.withdraw(address[],uint256[],bool) (contracts/core/Kernel.sol#459-479):
        External calls:
        - ethWithdrawn = IUserPositions(moduleMap.getModuleAddress(Modules.UserPositions)).withdraw(msg.sender,tokens,amounts,withdrawWethAsEth) (contracts/core/Kernel.sol#464-466)
        - IWeth9(ITIntegrationMap(moduleMap.getModuleAddress(Modules.IntegrationMap)).getWethTokenAddress()).withdraw(ethWithdrawn) (contracts/core/Kernel.sol#469-473)
        External calls sending eth:
        - address(msg.sender).transfer(ethWithdrawn) (contracts/core/Kernel.sol#475)
        Event emitted after the call(s):
        - Withdraw(msg.sender,tokens,amounts,ethWithdrawn) (contracts/core/Kernel.sol#478)
Reentrancy in Kernel.withdrawAllAndClaim(address[],bool) (contracts/core/Kernel.sol#488-525):
        External calls:
        - (tokenAmounts,ethWithdrawn,ethClaimed,biosClaimed) = IUserPositions(moduleMap.getModuleAddress(Modules.UserPositions)).withdrawAllAndClaim(msg.sender,tokens,withdrawWethAsEth) (contracts/core/Kernel.sol#500-502)
        - IWeth9(ITIntegrationMap(moduleMap.getModuleAddress(Modules.IntegrationMap)).getWethTokenAddress()).withdraw(ethWithdrawn) (contracts/core/Kernel.sol#505-509)
        External calls sending eth:
        - address(msg.sender).transfer(ethWithdrawn + ethClaimed) (contracts/core/Kernel.sol#513)
        Event emitted after the call(s):
        - WithdrawAllAndClaim(msg.sender,tokens,withdrawWethAsEth,tokenAmounts,ethWithdrawn,ethClaimed,biosClaimed) (contracts/core/Kernel.sol#516-524)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3

Kernel.getEthRewardsTimePeriod() (contracts/core/Kernel.sol#686-704) uses timestamp for comparisons
        Dangerous comparisons:
        - lastDistributeEthTimestamp > 0 (contracts/core/Kernel.sol#691)
        - lastLastDistributeEthTimestamp > 0 (contracts/core/Kernel.sol#692)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp

AddressUpgradeable.isContract(address) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#26-35) uses assembly
        - INLINE ASM (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#33)
AddressUpgradeable._verifyCallResult(bool,bytes,string) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#147-164) uses assembly
        - INLINE ASM (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#156-159)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

Different versions of Solidity is used:
        - Version used: ['0.8.4', '>=0.7.6', '^0.8.0']

        - ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/access/AccessControlEnumerableUpgradeable.sol#3)
        - ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/access/AccessControlUpgradeable.sol#3)
        - ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/proxy/utils/Initializable.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/security/ReentrancyGuardUpgradeable.sol#3)
        - ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/IERC20Upgradeable.sol#3)
        - ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/extensions/IERC20MetadataUpgradeable.sol#3)
        - ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/utils/SafeERC20Upgradeable.sol#3)
        - ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#3)
        - ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#3)
        - ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/utils/StringsUpgradeable.sol#3)
        - ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/utils/introspection/ERC165Upgradeable.sol#3)
        - ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/utils/introspection/IERC165Upgradeable.sol#3)
        - ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/utils/structs/EnumerableSetUpgradeable.sol#3)
        - 0.8.4 (contracts/core/Kernel.sol#2)
        - 0.8.4 (contracts/core/ModuleMapConsumer.sol#2)
        - 0.8.4 (contracts/interfaces/IBiosRewards.sol#2)
        - 0.8.4 (contracts/interfaces/IIntegration.sol#2)
        - 0.8.4 (contracts/interfaces/IIntegrationMap.sol#2)
        - 0.8.4 (contracts/interfaces/IInterconnects.sol#2)
        - 0.8.4 (contracts/interfaces/IKernel.sol#2)
        - 0.8.4 (contracts/interfaces/IModuleMap.sol#2)
        - 0.8.4 (contracts/interfaces/IStrategyMap.sol#2)
        - 0.8.4 (contracts/interfaces/ISushiSwapTrader.sol#2)
        - 0.8.4 (contracts/interfaces/IUniswapTrader.sol#2)
        - 0.8.4 (contracts/interfaces/IUserPositions.sol#2)
        - 0.8.4 (contracts/interfaces/IWeth9.sol#2)
        - 0.8.4 (contracts/interfaces/IYieldManager.sol#2)
        - >=0.7.6 (contracts/libraries/TokenSettings.sol#2)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used

AccessControlEnumerableUpgradeable.__AccessControlEnumerable_init() (node_modules/@openzeppelin/contracts-upgradeable/access/AccessControlEnumerableUpgradeable.sol#21-26) is never used and should be removed
AccessControlEnumerableUpgradeable.__AccessControlEnumerable_init_unchained() (node_modules/@openzeppelin/contracts-upgradeable/access/AccessControlEnumerableUpgradeable.sol#28-29) is never used and should be removed
AddressUpgradeable.functionCall(address,bytes) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#79-81) is never used and should be removed
AddressUpgradeable.functionCallWithValue(address,bytes,uint256) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#104-106) is never used and should be removed
AddressUpgradeable.functionStaticCall(address,bytes) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#129-131) is never used and should be removed
AddressUpgradeable.functionStaticCall(address,bytes,string) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#139-145) is never used and should be removed
AddressUpgradeable.sendValue(address,uint256) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#53-59) is never used and should be removed
ContextUpgradeable.__Context_init() (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#17-19) is never used and should be removed
ContextUpgradeable._msgData() (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#27-30) is never used and should be removed
ERC165Upgradeable.__ERC165_init() (node_modules/@openzeppelin/contracts-upgradeable/utils/introspection/ERC165Upgradeable.sol#23-25) is never used and should be removed
EnumerableSetUpgradeable.add(EnumerableSetUpgradeable.Bytes32Set,bytes32) (node_modules/@openzeppelin/contracts-upgradeable/utils/structs/EnumerableSetUpgradeable.sol#147-149) is never used and should be removed
```

```
EnumerableSetUpgradeable.add(EnumerableSetUpgradeable.UintSet,uint256) (node_modules/@openzeppelin/contracts-upgradeable/utils/structs/EnumerableSetUpgradeable.sol#256-258) is never used and should be removed
EnumerableSetUpgradeable.at(EnumerableSetUpgradeable.Bytes32Set,uint256) (node_modules/@openzeppelin/contracts-upgradeable/utils/structs/EnumerableSetUpgradeable.sol#185-187) is never used and should be removed
EnumerableSetUpgradeable.at(EnumerableSetUpgradeable.UintSet,uint256) (node_modules/@openzeppelin/contracts-upgradeable/utils/structs/EnumerableSetUpgradeable.sol#294-296) is never used and should be removed
EnumerableSetUpgradeable.contains(EnumerableSetUpgradeable.AddressSet,address) (node_modules/@openzeppelin/contracts-upgradeable/utils/structs/EnumerableSetUpgradeable.sol#218-220) is never used and should be removed
EnumerableSetUpgradeable.contains(EnumerableSetUpgradeable.Bytes32Set,bytes32) (node_modules/@openzeppelin/contracts-upgradeable/utils/structs/EnumerableSetUpgradeable.sol#164-166) is never used and should be removed
EnumerableSetUpgradeable.contains(EnumerableSetUpgradeable.UintSet,uint256) (node_modules/@openzeppelin/contracts-upgradeable/utils/structs/EnumerableSetUpgradeable.sol#273-275) is never used and should be removed
EnumerableSetUpgradeable.length(EnumerableSetUpgradeable.Bytes32Set) (node_modules/@openzeppelin/contracts-upgradeable/utils/structs/EnumerableSetUpgradeable.sol#171-173) is never used and should be removed
EnumerableSetUpgradeable.length(EnumerableSetUpgradeable.UintSet) (node_modules/@openzeppelin/contracts-upgradeable/utils/structs/EnumerableSetUpgradeable.sol#280-282) is never used and should be removed
EnumerableSetUpgradeable.remove(EnumerableSetUpgradeable.Bytes32Set,bytes32) (node_modules/@openzeppelin/contracts-upgradeable/utils/structs/EnumerableSetUpgradeable.sol#157-159) is never used and should be removed
EnumerableSetUpgradeable.remove(EnumerableSetUpgradeable.UintSet,uint256) (node_modules/@openzeppelin/contracts-upgradeable/utils/structs/EnumerableSetUpgradeable.sol#266-268) is never used and should be removed
SafeERC20Upgradeable.safeDecreaseAllowance(IERC20Upgradeable,address,uint256) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/utils/SafeERC20Upgradeable.sol#51-58) is never used and should be removed
SafeERC20Upgradeable.safeIncreaseAllowance(IERC20Upgradeable,address,uint256) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/utils/SafeERC20Upgradeable.sol#46-49) is never used and should be removed
SafeERC20Upgradeable.safeTransfer(IERC20Upgradeable,address,uint256) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/utils/SafeERC20Upgradeable.sol#20-22) is never used and should be removed
SafeERC20Upgradeable.safeTransferFrom(IERC20Upgradeable,address,address,uint256) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/utils/SafeERC20Upgradeable.sol#24-26) is never used and should be removed
StringsUpgradeable.toHexString(uint256) (node_modules/@openzeppelin/contracts-upgradeable/utils/StringsUpgradeable.sol#35-50) is never used and should be removed
StringsUpgradeable.toString(uint256) (node_modules/@openzeppelin/contracts-upgradeable/utils/StringsUpgradeable.sol#14-34) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/access/AccessControlEnumerableUpgradeable.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/access/AccessControlUpgradeable.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/proxy/utils/Initializable.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/security/ReentrancyGuardUpgradeable.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/IERC20Upgradeable.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/extensions/IERC20MetadataUpgradeable.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/utils/SafeERC20Upgradeable.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/utils/StringsUpgradeable.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/utils/introspection/ERC165Upgradeable.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/utils/introspection/IERC165Upgradeable.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/utils/structs/EnumerableSetUpgradeable.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version0.8.4 (contracts/core/Kernel.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version0.8.4 (contracts/core/ModuleMapConsumer.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version0.8.4 (contracts/interfaces/IBiosRewards.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version0.8.4 (contracts/interfaces/IIntegration.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version0.8.4 (contracts/interfaces/IIntegrationMap.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version0.8.4 (contracts/interfaces/IInterconnects.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version0.8.4 (contracts/interfaces/IKernel.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version0.8.4 (contracts/interfaces/IModuleMap.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version0.8.4 (contracts/interfaces/IStrategyMap.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version0.8.4 (contracts/interfaces/ISushiSwapTrader.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version0.8.4 (contracts/interfaces/IUniswapTrader.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version0.8.4 (contracts/interfaces/IUserPositions.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version0.8.4 (contracts/interfaces/IWeth9.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version0.8.4 (contracts/interfaces/IYieldManager.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
solc-0.8.4 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in AddressUpgradeable.sendValue(address,uint256) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#53-59):
        - (success) = recipient.call{value: amount}() (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#57)
Low level call in AddressUpgradeable.functionCallWithValue(address,bytes,uint256,string) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#114-121):
        - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#119)
Low level call in AddressUpgradeable.functionStaticCall(address,bytes,string) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#139-145):
        - (success,returndata) = target.staticcall(data) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#143)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Function AccessControlEnumerableUpgradeable.__AccessControlEnumerable_init() (node_modules/@openzeppelin/contracts-upgradeable/access/AccessControlEnumerableUpgradeable.sol#21-26) is not in mixedCase
Function AccessControlEnumerableUpgradeable.__AccessControlEnumerable_init_unchained() (node_modules/@openzeppelin/contracts-upgradeable/access/AccessControlEnumerableUpgradeable.sol#28-29) is not in mixedCase
Variable AccessControlEnumerableUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/access/AccessControlEnumerableUpgradeable.sol#97) is not in mixedCase
Function AccessControlUpgradeable._AccessControl_init() (node_modules/@openzeppelin/contracts-upgradeable/access/AccessControlUpgradeable.sol#60-64) is not in mixedCase
Function AccessControlUpgradeable._AccessControl_init_unchained() (node_modules/@openzeppelin/contracts-upgradeable/access/AccessControlUpgradeable.sol#66-67) is not in mixedCase
Variable AccessControlUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/access/AccessControlUpgradeable.sol#252) is not in mixedCase
Function ReentrancyGuardUpgradeable.__ReentrancyGuard_init() (node_modules/@openzeppelin/contracts-upgradeable/security/ReentrancyGuardUpgradeable.sol#38-41) is not in mixedCase
Function ReentrancyGuardUpgradeable.__ReentrancyGuard_init_unchained() (node_modules/@openzeppelin/contracts-upgradeable/security/ReentrancyGuardUpgradeable.sol#43-45) is not in mixedCase
Variable ReentrancyGuardUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/security/ReentrancyGuardUpgradeable.sol#67) is not in mixedCase
Function ContextUpgradeable._Context_init() (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#17-18) is not in mixedCase
Function ContextUpgradeable._Context_init_unchained() (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#21-22) is not in mixedCase
Variable ContextUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#31) is not in mixedCase
Constant StringsUpgradeable.alphabet (node_modules/@openzeppelin/contracts-upgradeable/utils/StringsUpgradeable.sol#9) is not in UPPER_CASE_WITH_UNDERSCORES
Function ERC165Upgradeable.__ERC165_init() (node_modules/@openzeppelin/contracts-upgradeable/utils/introspection/ERC165Upgradeable.sol#23-25) is not in mixedCase
Function ERC165Upgradeable.__ERC165_init_unchained() (node_modules/@openzeppelin/contracts-upgradeable/utils/introspection/ERC165Upgradeable.sol#27-28) is not in mixedCase
Variable ERC165Upgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/utils/introspection/ERC165Upgradeable.sol#35) is not in mixedCase
Function ModuleMapConsumer.__ModuleMapConsumer_init(address) (contracts/core/ModuleMapConsumer.sol#10-12) is not in mixedCase
Struct IInterconnects.lpData (contracts/interfaces/IInterconnects.sol#6-9) is not in CapWords
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Redundant expression "this (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#28)" inContextUpgradeable (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#16-32)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements

Reentrancy in Kernel.claimAllRewards() (contracts/core/Kernel.sol#593-600):
        External calls:
        - ethClaimed = claimEthRewards() (contracts/core/Kernel.sol#598)
                - address(msg.sender).transfer(ethClaimed) (contracts/core/Kernel.sol#571)
        State variables written after the call(s):
        - biosClaimed = claimBiosRewards() (contracts/core/Kernel.sol#599)
                - _status = _ENTERED (node_modules/@openzeppelin/contracts-upgradeable/security/ReentrancyGuardUpgradeable.sol#59)
                - _status = _NOT_ENTERED (node_modules/@openzeppelin/contracts-upgradeable/security/ReentrancyGuardUpgradeable.sol#65)
        Event emitted after the call(s):
        - ClaimBiosRewards(msg.sender,biosClaimed) (contracts/core/Kernel.sol#587)
                - biosClaimed = claimBiosRewards() (contracts/core/Kernel.sol#599)
Reentrancy in Kernel.claimEthRewards() (contracts/core/Kernel.sol#562-574):
        External calls:
        - address(msg.sender).transfer(ethClaimed) (contracts/core/Kernel.sol#571)
        Event emitted after the call(s):
        - ClaimEthRewards(msg.sender,ethClaimed) (contracts/core/Kernel.sol#573)
Reentrancy in Kernel.withdraw(address[],uint256[],bool) (contracts/core/Kernel.sol#459-479):
        External calls:
        - address(msg.sender).transfer(ethWithdrawn) (contracts/core/Kernel.sol#475)
        Event emitted after the call(s):
        - Withdraw(msg.sender,tokens,amounts,ethWithdrawn) (contracts/core/Kernel.sol#478)
Reentrancy in Kernel.withdrawAllAndClaim(address[],bool) (contracts/core/Kernel.sol#488-525):
        External calls:
        - address(msg.sender).transfer(ethWithdrawn + ethClaimed) (contracts/core/Kernel.sol#513)
        Event emitted after the call(s):
        - WithdrawAllAndClaim(msg.sender,tokens,withdrawnWethAsEth,tokenAmounts,ethWithdrawn,ethClaimed,biosClaimed) (contracts/core/Kernel.sol#516-524)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-4

ReentrancyGuardUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/security/ReentrancyGuardUpgradeable.sol#67) is never used in Kernel (contracts/core/Kernel.sol#24-709)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variable

getRoleMember(bytes32,uint256) should be declared external:
        - AccessControlEnumerableUpgradeable.getRoleMember(bytes32,uint256) (node_modules/@openzeppelin/contracts-upgradeable/access/AccessControlEnumerableUpgradeable.sol#54-56)
getRoleMemberCount(bytes32) should be declared external:
        - AccessControlEnumerableUpgradeable.getRoleMemberCount(bytes32) (node_modules/@openzeppelin/contracts-upgradeable/access/AccessControlEnumerableUpgradeable.sol#62-64)
isManager(address) should be declared external:
        - Kernel.isManager(address) (contracts/core/Kernel.sol#644-646)
isOwner(address) should be declared external:
        - Kernel.isOwner(address) (contracts/core/Kernel.sol#650-652)
isLiquidityProvider(address) should be declared external:
        - Kernel.isLiquidityProvider(address) (contracts/core/Kernel.sol#656-663)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

IntegrationMap.sol

```
Controlled.__Controlled_init(address[],address).i (contracts/core/Controlled.sol#17) is a local variable never initialized
IntegrationMap.getBiosRewardWeightSum().tokenId (contracts/core/IntegrationMap.sol#540) is a local variable never initialized
Controlled.addController(address).i (contracts/core/Controlled.sol#27) is a local variable never initialized
Controlled.addController(address).added (contracts/core/Controlled.sol#26) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables

IntegrationMap.initialize(address[],address,address,address).wethTokenAddress_ (contracts/core/IntegrationMap.sol#33) lacks a zero-check on :
        - wethTokenAddress = wethTokenAddress_ (contracts/core/IntegrationMap.sol#37)
IntegrationMap.initialize(address[],address,address,address).biosTokenAddress_ (contracts/core/IntegrationMap.sol#34) lacks a zero-check on :
        - biosTokenAddress = biosTokenAddress_ (contracts/core/IntegrationMap.sol#38)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

Different versions of Solidity is used:
        - Version used: ['0.8.4', '>=0.7.6', '^0.8.0']
        - ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/proxy/utils/Initializable.sol#4)
        - 0.8.4 (contracts/core/Controlled.sol#2)
        - 0.8.4 (contracts/core/IntegrationMap.sol#2)
        - 0.8.4 (contracts/core/ModuleMapConsumer.sol#2)
        - 0.8.4 (contracts/interfaces/IIntegrationMap.sol#2)
        - 0.8.4 (contracts/interfaces/IKernel.sol#2)
        - 0.8.4 (contracts/interfaces/IModuleMap.sol#2)
        - >=0.7.6 (contracts/libraries/TokenSettings.sol#2)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used

Pragma version^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/proxy/utils/Initializable.sol#4) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version0.8.4 (contracts/core/Controlled.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version0.8.4 (contracts/core/IntegrationMap.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version0.8.4 (contracts/core/ModuleMapConsumer.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version0.8.4 (contracts/interfaces/IIntegrationMap.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version0.8.4 (contracts/interfaces/IKernel.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version0.8.4 (contracts/interfaces/IModuleMap.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
solc-0.8.4 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Function Controlled.__Controlled_init(address[],address) (contracts/core/Controlled.sol#13-22) is not in mixedCase
Variable Controlled._controllers (contracts/core/Controlled.sol#10) is not in mixedCase
Function ModuleMapConsumer.__ModuleMapConsumer_init(address) (contracts/core/ModuleMapConsumer.sol#10-12) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Variable Controlled._controllers (contracts/core/Controlled.sol#10) is too similar to Controlled.__Controlled_init(address[],address).controllers_ (contracts/core/Controlled.sol#14)
Variable Controlled._controllers (contracts/core/Controlled.sol#10) is too similar to IntegrationMap.initialize(address[],address,address,address).controllers_ (contracts/core/IntegrationMap.sol#31)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar

initialize(address[],address,address,address) should be declared external:
        - IntegrationMap.initialize(address[],address,address,address) (contracts/core/IntegrationMap.sol#30-64)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

AUTOMATED TESTING

Interconnects.sol

```
Interconnects.calculateBridgeFeesAndNetAmount(address[],uint256[]) (contracts/core/Interconnects.sol#401-448) performs a multiplication on the result of a division:
        -x = (_tokenPoolLpBalances[token] * targetLiquidityRatioDenominator) / (integrationMap.getTokenTargetLiquidityRatioNumerator(token) * getTokenPoolLpActiveFositions(token)) (contracts/core/Interconnects.sol#431-434)
        -totalFee = (integrationMap.getTokenTransferFeeKValueNumerator(token) * amounts[tokenId]) / (kValueDenominator * x) (contracts/core/Interconnects.sol#436-438)
Interconnects.calculateBridgeFeesAndNetAmount(address[],uint256[]) (contracts/core/Interconnects.sol#401-448) performs a multiplication on the result of a division:
        -totalFee = (integrationMap.getTokenTransferFeeKValueNumerator(token) * amounts[tokenId]) / (kValueDenominator * x) (contracts/core/Interconnects.sol#436-438)
        -protocolFees[tokenId] = (totalFee * protocolFeeNumerator) / protocolFeeDenominator (contracts/core/Interconnects.sol#440-442)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply

Reentrancy in Interconnects.bridgeTo(address,address[],uint256[]) (contracts/core/Interconnects.sol#297-337):
        External calls:
        - userPositions.updateUserInterconnectBalances(tokens,user,netBridgeAmounts,addArr) (contracts/core/Interconnects.sol#323-328)
        State variables written after the call(s):
        - updateTokenUserLpFeeRewardBalances(token,lpFee) (contracts/core/Interconnects.sol#333)
                - _tokenUserLpBalances[asset][lpUser].rewards += (lpTransferFeeAmt * _tokenUserLpBalances[asset][lpUser].position) / totalAmountAllocated (contracts/core/Interconnects.sol#392-395)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1

Interconnects.claimLpFeeRewards(address,address[]).tokenId (contracts/core/Interconnects.sol#226) is a local variable never initialized
Interconnects.takeLiquidity(address,address[],uint256[]).lpUserIdx (contracts/core/Interconnects.sol#188) is a local variable never initialized
Interconnects.undoBridgeFrom(address,address[],uint256[]).i (contracts/core/Interconnects.sol#283) is a local variable never initialized
Interconnects.calculateBridgeFeesAndNetAmount(address[],uint256[]).tokenId (contracts/core/Interconnects.sol#425) is a local variable never initialized
Interconnects.bridgeTo(address,address[],uint256[]).i (contracts/core/Interconnects.sol#311) is a local variable never initialized
Interconnects.takeLiquidity(address,address[],uint256[]).tokenId (contracts/core/Interconnects.sol#164) is a local variable never initialized
Controlled.addController(address).i (contracts/core/Controlled.sol#27) is a local variable never initialized
Controlled.addController(address).added (contracts/core/Controlled.sol#26) is a local variable never initialized
Interconnects.updateTokenPoolLpBalances(address[],uint256[],bool[]).tokenId (contracts/core/Interconnects.sol#356) is a local variable never initialized
Interconnects.provideLiquidity(address,address[],uint256[]).tokenId (contracts/core/Interconnects.sol#111) is a local variable never initialized
Interconnects.updateTokenUserLpFeeRewardBalances(address,uint256).lpUserIdx (contracts/core/Interconnects.sol#386) is a local variable never initialized
Interconnects.bridgeTo(address,address[],uint256[]).i_scope_0 (contracts/core/Interconnects.sol#329) is a local variable never initialized
Controlled._Controlled_init(address[],address).i (contracts/core/Controlled.sol#17) is a local variable never initialized
Interconnects.getTokenPoolLpActiveFositions(address).lpUserIdx (contracts/core/Interconnects.sol#451) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables

Interconnects.initialize(address[],address,address).relayAccount_ (contracts/core/Interconnects.sol#40) lacks a zero-check on :
                - _relayAccount = relayAccount_ (contracts/core/Interconnects.sol#43)
Interconnects.updateRelayAccounts(address).relayAccount_ (contracts/core/Interconnects.sol#75) lacks a zero-check on :
                - _relayAccount = relayAccount_ (contracts/core/Interconnects.sol#80)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

Interconnects.provideLiquidity(address,address[],uint256[]) (contracts/core/Interconnects.sol#92-142) has external calls inside a loop: require(bool,string)(integrationMap.getTokenAcceptingLping(tokens[tokenId]),Interconnects::provideLiquidity: This token is not accepting LP positions) (contracts/core/Interconnects.sol#112-115)
Interconnects.provideLiquidity(address,address[],uint256[]) (contracts/core/Interconnects.sol#92-142) has external calls inside a loop: require(bool,string)(amounts[tokenId] > 0 && amounts[tokenId] <= userPositions.userTokenBalance(token,user),Interconnects::provideLiquidity: LP amount must be greater than zero and no more than the user's balance) (contracts/core/Interconnects.sol#117-122)
Interconnects.updateTokenPoolLpBalances(address[],uint256[],bool[]) (contracts/core/Interconnects.sol#342-376) has external calls inside a loop: require(bool,string)(integrationMap.getTokenAcceptingLping(tokens[tokenId]),Interconnects::updateTokenPoolLpBalances: This token is not accepting LP positions) (contracts/core/Interconnects.sol#357-360)
Interconnects.calculateBridgeFeesAndNetAmount(address[],uint256[]) (contracts/core/Interconnects.sol#401-448) has external calls inside a loop: protocolFeeNumerator = integrationMap.getTokenTransferFeePlatformRatioNumerator(token) (contracts/core/Interconnects.sol#427-428)
Interconnects.calculateBridgeFeesAndNetAmount(address[],uint256[]) (contracts/core/Interconnects.sol#401-448) has external calls inside a loop: x = (_tokenPoolLpBalances[token] * targetLiquidityRatioDenominator) / (integrationMap.getTokenTargetLiquidityRatioNumerator(token) * getTokenPoolLpActiveFositions(token)) (contracts/core/Interconnects.sol#431-434)
Interconnects.calculateBridgeFeesAndNetAmount(address[],uint256[]) (contracts/core/Interconnects.sol#401-448) has external calls inside a loop: totalFee = (integrationMap.getTokenTransferFeeKValueNumerator(token) * amounts[tokenId]) / (kValueDenominator * x) (contracts/core/Interconnects.sol#436-438)
Interconnects.validateBridging(address[],uint256[]) (contracts/core/Interconnects.sol#54-72) has external calls inside a loop: require(bool,string)(integrationMap.getTokenAcceptingBridging(tokens[i]),Interconnects::validateBridging: Token is not accepting bridging) (contracts/core/Interconnects.sol#66-69)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-inside-a-loop

Reentrancy in Interconnects.bridgeTo(address,address[],uint256[]) (contracts/core/Interconnects.sol#297-337):
        External calls:
        - userPositions.updateUserInterconnectBalances(tokens,user,netBridgeAmounts,addArr) (contracts/core/Interconnects.sol#323-328)
        State variables written after the call(s):
        - _tokenProtocolFeeRewards[token] += protocolFee (contracts/core/Interconnects.sol#334)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2

Reentrancy in Interconnects.bridgeFrom(address,address[],uint256[]) (contracts/core/Interconnects.sol#246-266):
        External calls:
        - userPositions.updateUserTokenBalances(tokens,user,amounts,addArr) (contracts/core/Interconnects.sol#262)
        Event emitted after the call(s):
        - BridgeFrom(user,tokens,amounts) (contracts/core/Interconnects.sol#263)
Reentrancy in Interconnects.bridgeTo(address,address[],uint256[]) (contracts/core/Interconnects.sol#297-337):
        External calls:
        - userPositions.updateUserInterconnectBalances(tokens,user,netBridgeAmounts,addArr) (contracts/core/Interconnects.sol#323-328)
        Event emitted after the call(s):
        - BridgeTo(user,tokens,amounts) (contracts/core/Interconnects.sol#336)
Reentrancy in Interconnects.claimLpFeeRewards(address,address[]) (contracts/core/Interconnects.sol#214-240):
        External calls:
        - userPositions.updateUserTokenBalances(tokens,user,amounts,add) (contracts/core/Interconnects.sol#237)
        Event emitted after the call(s):
        - ClaimLpFeeRewards(user,tokens) (contracts/core/Interconnects.sol#239)
Reentrancy in Interconnects.provideLiquidity(address,address[],uint256[]) (contracts/core/Interconnects.sol#92-142):
        External calls:
        - userPositions.updateUserTokenBalances(tokenArr,user,amountArr,addArr) (contracts/core/Interconnects.sol#135-140)
        Event emitted after the call(s):
        - ProvideLiquidity(user,tokens,amounts) (contracts/core/Interconnects.sol#141)
Reentrancy in Interconnects.takeLiquidity(address,address[],uint256[]) (contracts/core/Interconnects.sol#147-209):
        External calls:
        - userPositions.updateUserTokenBalances(tokenArr,user,amountArr,addArr) (contracts/core/Interconnects.sol#202-207)
        Event emitted after the call(s):
        - TakeLiquidity(user,tokens,amounts) (contracts/core/Interconnects.sol#208)
Reentrancy in Interconnects.undoBridgeFrom(address,address[],uint256[]) (contracts/core/Interconnects.sol#270-291):
        External calls:
        - userPositions.updateUserTokenBalances(tokens,user,amounts,addArr) (contracts/core/Interconnects.sol#289)
        Event emitted after the call(s):
        - UndoBridgeFrom(user,tokens,amounts) (contracts/core/Interconnects.sol#290)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3

Different versions of Solidity is used:
        - Version used: ['0.8.4', '>=0.7.6', '^0.8.0']
        - ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/proxy/utils/Initializable.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/security/ReentrancyGuardUpgradeable.sol#3)
        - 0.8.4 (contracts/core/Controlled.sol#2)
        - 0.8.4 (contracts/core/Interconnects.sol#2)
        - 0.8.4 (contracts/core/ModuleMapConsumer.sol#2)
        - 0.8.4 (contracts/interfaces/IIntegration.sol#2)
        - 0.8.4 (contracts/interfaces/IIntegrationMap.sol#2)
        - 0.8.4 (contracts/interfaces/IInterconnects.sol#2)
        - 0.8.4 (contracts/interfaces/IKernel.sol#2)
        - 0.8.4 (contracts/interfaces/IModuleMap.sol#2)
        - 0.8.4 (contracts/interfaces/IStrategyMap.sol#2)
        - 0.8.4 (contracts/interfaces/IUserPositions.sol#2)
        - >=0.7.6 (contracts/libraries/TokenSettings.sol#2)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used

ReentrancyGuardUpgradeable.__ReentrancyGuard_init() (node_modules/@openzeppelin/contracts-upgradeable/security/ReentrancyGuardUpgradeable.sol#39-41) is never used and should be removed
ReentrancyGuardUpgradeable.__ReentrancyGuard_init_unchained() (node_modules/@openzeppelin/contracts-upgradeable/security/ReentrancyGuardUpgradeable.sol#43-45) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/proxy/utils/Initializable.sol#4) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/security/ReentrancyGuardUpgradeable.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version0.8.4 (contracts/core/Controlled.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version0.8.4 (contracts/core/Interconnects.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version0.8.4 (contracts/core/ModuleMapConsumer.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version0.8.4 (contracts/interfaces/IIntegration.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version0.8.4 (contracts/interfaces/IIntegrationMap.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version0.8.4 (contracts/interfaces/IInterconnects.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version0.8.4 (contracts/interfaces/IKernel.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version0.8.4 (contracts/interfaces/IModuleMap.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version0.8.4 (contracts/interfaces/IStrategyMap.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version0.8.4 (contracts/interfaces/IUserPositions.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
solc-0.8.4 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Function ReentrancyGuardUpgradeable.__ReentrancyGuard_init() (node_modules/@openzeppelin/contracts-upgradeable/security/ReentrancyGuardUpgradeable.sol#39-41) is not in mixedCase
Function ReentrancyGuardUpgradeable.__ReentrancyGuard_init_unchained() (node_modules/@openzeppelin/contracts-upgradeable/security/ReentrancyGuardUpgradeable.sol#43-45) is not in mixedCase
Variable ReentrancyGuardUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/security/ReentrancyGuardUpgradeable.sol#67) is not in mixedCase
Function Controlled._Controlled_init(address[],address) (contracts/core/Controlled.sol#13-22) is not in mixedCase
Variable Controlled._controllers (contracts/core/Controlled.sol#10) is not in mixedCase
Function ModuleMapConsumer.__ModuleMapConsumer_init(address) (contracts/core/ModuleMapConsumer.sol#10-12) is not in mixedCase
Struct IInterconnects.lpData (contracts/interfaces/IInterconnects.sol#6-9) is not in CapWords
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Variable Controlled._controllers (contracts/core/Controlled.sol#10) is too similar to Controlled._Controlled_init(address[],address).controllers_ (contracts/core/Controlled.sol#14)
Variable Controlled._controllers (contracts/core/Controlled.sol#10) is too similar to Interconnects.initialize(address[],address,address).controllers_ (contracts/core/Interconnects.sol#38)
Variable Interconnects._relayAccount (contracts/core/Interconnects.sol#20) is too similar to Interconnects.initialize(address[],address,address).relayAccount_ (contracts/core/Interconnects.sol#40)
Variable Interconnects._relayAccount (contracts/core/Interconnects.sol#20) is too similar to Interconnects.updateRelayAccount(address).relayAccount_ (contracts/core/Interconnects.sol#75)
Variable Interconnects._relayAccount (contracts/core/Interconnects.sol#20) is too similar to IInterconnects.updateRelayAccount(address).relayAccount_ (contracts/interfaces/IInterconnects.sol#42)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar

ReentrancyGuardUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/security/ReentrancyGuardUpgradeable.sol#67) is never used in Interconnects (contracts/core/Interconnects.sol#12-518)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variable

initialize(address[],address,address) should be declared external:
        - Interconnects.initialize(address[],address,address) (contracts/core/Interconnects.sol#37-44)
getRelayAccount() should be declared external:
        - Interconnects.getRelayAccount() (contracts/core/Interconnects.sol#84-86)
getTokenUserLpBalance(address,address) should be declared external:
        - Interconnects.getTokenUserLpBalance(address,address) (contracts/core/Interconnects.sol#452-459)
getTokenPoolLpBalance(address) should be declared external:
        - Interconnects.getTokenPoolLpBalance(address) (contracts/core/Interconnects.sol#462-469)
getTokenUserLpFeeRewardBalance(address,address) should be declared external:
        - Interconnects.getTokenUserLpFeeRewardBalance(address,address) (contracts/core/Interconnects.sol#490-497)
getTokenLpUsers(address) should be declared external:
        - Interconnects.getTokenLpUsers(address) (contracts/core/Interconnects.sol#500-507)
getTokenProtocolFeeRewards(address) should be declared external:
        - Interconnects.getTokenProtocolFeeRewards(address) (contracts/core/Interconnects.sol#510-517)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

YieldManager.sol

```
YieldManager.distributeEth() (contracts/core/YieldManager.sol#328-369) sends eth to arbitrary user
        Dangerous calls:
        - address(treasuryAccount).transfer(treasuryWethAmount) (contracts/core/YieldManager.sol#361)
YieldManager.ethToGasAccount() (contracts/core/YieldManager.sol#372-403) sends eth to arbitrary user
        Dangerous calls:
        - gasAccount.transfer(ethAmountToGasAccount) (contracts/core/YieldManager.sol#392)
        - gasAccount.transfer(ethAmountToGasAccount) (contracts/core/YieldManager.sol#399)
YieldManager.ethToRewards(uint256) (contracts/core/YieldManager.sol#427-463) sends eth to arbitrary user
        Dangerous calls:
        - address(moduleMap.getModuleAddress(Modules.Kernel)).transfer(ethRewardsAmount) (contracts/core/YieldManager.sol#460-462)
YieldManager.ethToProtocolFeeAccrual(uint256) (contracts/core/YieldManager.sol#467-496) sends eth to arbitrary user
        Dangerous calls:
        - address(moduleMap.getModuleAddress(Modules.Kernel)).transfer(protocolFeeEthRewardsAmount) (contracts/core/YieldManager.sol#486-488)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#functions-that-send-ether-to-arbitrary-destinations

YieldManager.ethToProtocolFeeAccrual(uint256) (contracts/core/YieldManager.sol#467-496) ignores return value by IERC20MetadataUpgradeable(wethAddress).transfer(moduleMap.getModuleAddress(Modules.Kernel),protocolFeeEthRewardsAmount) (contracts/core/YieldManager.sol#491-494)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-transfer

YieldManager._calculateReserveAmount(uint256,uint256,uint256) (contracts/core/YieldManager.sol#237-243) uses a dangerous strict equality:
        - amount == 0 (contracts/core/YieldManager.sol#242)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities

Reentrancy in YieldManager.ethToRewards(uint256) (contracts/core/YieldManager.sol#427-463):
        External calls:
        - IEtherRewards(moduleMap.getModuleAddress(Modules.EtherRewards)).increaseEthRewards(tokenAddress,(ethRewardsAmount * processedWethByToken[tokenAddress]) / processedWethByTokenSum) (contracts/core/YieldManager.sol#444-450)
        State variables written after the call(s):
        - processedWethByToken[tokenAddress] = 0 (contracts/core/YieldManager.sol#452)
Reentrancy in YieldManager.harvestYield(address,address[]) (contracts/core/YieldManager.sol#246-325):
        External calls:
        - IIntegration(integrationAddress).harvestYield() (contracts/core/YieldManager.sol#262)
        - token.safeTransfer(moduleMap.getModuleAddress(Modules.Kernel),_calculateReserveAmount(token.balanceOf(address(this)),integrationMap.getTokenReserveRatioNumerator(address(token)),integrationMap.getReserveRatioDenominator())) (contracts/core/YieldManager.sol#278-288)
        - token.safeTransfer(moduleMap.getModuleAddress(Modules.SwapManager),token.balanceOf(address(this))) (contracts/core/YieldManager.sol#293-296)
        - ISwapManager(moduleMap.getModuleAddress(Modules.SwapManager)).swapExactIn(address(token),address(weth),address(this),token.balanceOf(moduleMap.getModuleAddress(Modules.SwapManager))) (contracts/core/YieldManager.sol#298-309)
        State variables written after the call(s):
        - processedWethByToken[address(token)] += weth.balanceOf(address(this)) - wethBalanceBefore (contracts/core/YieldManager.sol#311-313)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1

YieldManager.ethToRewards(uint256).tokenId (contracts/core/YieldManager.sol#440) is a local variable never initialized
YieldManager.harvestYield(address,address[]).tokenIterator (contracts/core/YieldManager.sol#270) is a local variable never initialized
Controlled._Controlled_init(address[],address).i (contracts/core/Controlled.sol#17) is a local variable never initialized
YieldManager.getProcessedWethByTokenSum(address[]).tokenId (contracts/core/YieldManager.sol#553) is a local variable never initialized
Controlled.addController(address).i (contracts/core/Controlled.sol#27) is a local variable never initialized
YieldManager.getAllPendingYield(address).integrationId (contracts/core/YieldManager.sol#704) is a local variable never initialized
YieldManager.getProcessedWethSum().tokenId (contracts/core/YieldManager.sol#567) is a local variable never initialized
YieldManager.getTokenTotalIntegrationBalance(address).integrationId (contracts/core/YieldManager.sol#615) is a local variable never initialized
Controlled.addController(address).added (contracts/core/Controlled.sol#26) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables

YieldManager.harvestYield(address,address[]) (contracts/core/YieldManager.sol#246-325) ignores return value by ISwapManager(moduleMap.getModuleAddress(Modules.SwapManager)).swapExactIn(address(token),address(weth),address(this),token.balanceOf(moduleMap.getModuleAddress(Modules.SwapManager))) (contracts/core/YieldManager.sol#298-309)
YieldManager.bioxBuyBack() (contracts/core/YieldManager.sol#406-423) ignores return value by ISushiSwapTrader(moduleMap.getModuleAddress(Modules.SushiSwapTrader)).bioxBuyBack() (contracts/core/YieldManager.sol#415-417)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return

YieldManager.initialize(address[],address,uint256,uint32,uint32,uint32,uint32,address,address).gasAccount_ (contracts/core/YieldManager.sol#69) lacks a zero-check on :
                - gasAccount = gasAccount_ (contracts/core/YieldManager.sol#78)
YieldManager.initialize(address[],address,uint256,uint32,uint32,uint32,uint32,address,address).treasuryAccount_ (contracts/core/YieldManager.sol#70) lacks a zero-check on :
                - treasuryAccount = treasuryAccount_ (contracts/core/YieldManager.sol#79)
YieldManager.updateGasAccount(address).gasAccount_ (contracts/core/YieldManager.sol#106) lacks a zero-check on :
                - gasAccount = gasAccount_ (contracts/core/YieldManager.sol#111)
YieldManager.updateTreasuryAccount(address).treasuryAccount_ (contracts/core/YieldManager.sol#115) lacks a zero-check on :
                - treasuryAccount = treasuryAccount_ (contracts/core/YieldManager.sol#120)

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

YieldManager.deploy(ITYieldManager.DeployRequest[]) (contracts/core/YieldManager.sol#124-231) has external calls inside a loop: deployAmount = strategyMap.getDeployAmount(deployments[i].integration,deployments[i].ammFoolID,deployments[i].tokens[j]) (contracts/core/YieldManager.sol#139-143)
YieldManager.deploy(ITYieldManager.DeployRequest[]) (contracts/core/YieldManager.sol#124-231) has external calls inside a loop: reserveBalance = IERC20MetadataUpgradeable(deployments[i].tokens[j]).balanceOf(moduleMap.getModuleAddress(Modules.Kernel)) (contracts/core/YieldManager.sol#145-147)
YieldManager.deploy(ITYieldManager.DeployRequest[]) (contracts/core/YieldManager.sol#124-231) has external calls inside a loop: strategyMap.closePositionsForWithdrawal(deployments[i].tokens[j],type()(uint256).max) (contracts/core/YieldManager.sol#149-152)
YieldManager.deploy(ITYieldManager.DeployRequest[]) (contracts/core/YieldManager.sol#124-231) has external calls inside a loop: deployAmount = strategyMap.getDeployAmount(deployments[i].integration,deployments[i].ammFoolID,deployments[i].tokens[j]) (contracts/core/YieldManager.sol#153-157)
YieldManager.deploy(ITYieldManager.DeployRequest[]) (contracts/core/YieldManager.sol#124-231) has external calls inside a loop: balanceBefore = IERC20MetadataUpgradeable(deployments[i].tokens[j]).balanceOf(deployments[i].integration) (contracts/core/YieldManager.sol#166-168)
YieldManager.deploy(ITYieldManager.DeployRequest[]) (contracts/core/YieldManager.sol#124-231) has external calls inside a loop: IERC20MetadataUpgradeable(deployments[i].tokens[j]).safeTransferFrom(moduleMap.getModuleAddress(Modules.Kernel),deployments[i].integration,abs(deployAmount)) (contracts/core/YieldManager.sol#170-175)
YieldManager.deploy(ITYieldManager.DeployRequest[]) (contracts/core/YieldManager.sol#124-231) has external calls inside a loop: balanceAfter = IERC20MetadataUpgradeable(deployments[i].tokens[j]).balanceOf(deployments[i].integration) (contracts/core/YieldManager.sol#176-178)
YieldManager.deploy(ITYieldManager.DeployRequest[]) (contracts/core/YieldManager.sol#124-231) has external calls inside a loop: integration.deposit(deployments[i].tokens[j],balanceAfter - balanceBefore,deployments[i].ammFoolID) (contracts/core/YieldManager.sol#179-183)
YieldManager.deploy(ITYieldManager.DeployRequest[]) (contracts/core/YieldManager.sol#124-231) has external calls inside a loop: integration.deploy(deployments[i].ammFoolID) (contracts/core/YieldManager.sol#184)
YieldManager.deploy(ITYieldManager.DeployRequest[]) (contracts/core/YieldManager.sol#124-231) has external calls inside a loop: strategyMap.decreaseDeployAmountChange(deployments[i].integration,deployments[i].ammFoolID,deployments[i].tokens[j],abs(deployAmount)) (contracts/core/YieldManager.sol#222-227)
YieldManager.deploy(ITYieldManager.DeployRequest[]) (contracts/core/YieldManager.sol#124-231) has external calls inside a loop: integration.withdraw(deployments[i].tokens[j],abs(deployAmount),deployments[i].ammFoolID) (contracts/core/YieldManager.sol#186-190)
YieldManager.deploy(ITYieldManager.DeployRequest[]) (contracts/core/YieldManager.sol#124-231) has external calls inside a loop: balanceBefore_scope_1 = IERC20MetadataUpgradeable(deployments[i].tokens[j]).balanceOf(deployments[i].integration) (contracts/core/YieldManager.sol#197-199)
YieldManager.deploy(ITYieldManager.DeployRequest[]) (contracts/core/YieldManager.sol#124-231) has external calls inside a loop: IERC20MetadataUpgradeable(deployments[i].tokens[j]).safeTransferFrom(moduleMap.getModuleAddress(Modules.Kernel),deployments[i].integration,abs(deployAmount)) (contracts/core/YieldManager.sol#200-205)
YieldManager.deploy(ITYieldManager.DeployRequest[]) (contracts/core/YieldManager.sol#124-231) has external calls inside a loop: balanceAfter_scope_2 = IERC20MetadataUpgradeable(deployments[i].tokens[j]).balanceOf(deployments[i].integration) (contracts/core/YieldManager.sol#206-208)
YieldManager.deploy(ITYieldManager.DeployRequest[]) (contracts/core/YieldManager.sol#124-231) has external calls inside a loop: integration_scope_0.deposit(deployments[i].tokens[j],balanceAfter_scope_2 - balanceBefore_scope_1) (contracts/core/YieldManager.sol#210-213)
YieldManager.deploy(ITYieldManager.DeployRequest[]) (contracts/core/YieldManager.sol#124-231) has external calls inside a loop: integration_scope_0.deploy() (contracts/core/YieldManager.sol#214)
YieldManager.deploy(ITYieldManager.DeployRequest[]) (contracts/core/YieldManager.sol#124-231) has external calls inside a loop: integration_scope_0.withdraw(deployments[i].tokens[j],abs(deployAmount)) (contracts/core/YieldManager.sol#216-219)
YieldManager.harvestYield(address,address[]) (contracts/core/YieldManager.sol#246-325) has external calls inside a loop: token.balanceOf(address(this)) > 0 (contracts/core/YieldManager.sol#278)
YieldManager.harvestYield(address,address[]) (contracts/core/YieldManager.sol#246-325) has external calls inside a loop: token.safeTransfer(moduleMap.getModuleAddress(Modules.Kernel),_calculateReserveAmount(token.balanceOf(address(this)),integrationMap.getTokenReserveRatioNumerator(address(token)),integrationMap.getReserveRatioDenominator())) (contracts/core/YieldManager.sol#278-288)
YieldManager.harvestYield(address,address[]) (contracts/core/YieldManager.sol#246-325) has external calls inside a loop: wethBalanceBefore = weth.balanceOf(address(this)) (contracts/core/YieldManager.sol#290)
YieldManager.harvestYield(address,address[]) (contracts/core/YieldManager.sol#246-325) has external calls inside a loop: token.safeTransfer(moduleMap.getModuleAddress(Modules.SwapManager),token.balanceOf(address(this))) (contracts/core/YieldManager.sol#293-296)
YieldManager.harvestYield(address,address[]) (contracts/core/YieldManager.sol#246-325) has external calls inside a loop: ISwapManager(moduleMap.getModuleAddress(Modules.SwapManager)).swapExactIn(address(token),address(weth),address(this),token.balanceOf(moduleMap.getModuleAddress(Modules.SwapManager))) (contracts/core/YieldManager.sol#298-309)
YieldManager.harvestYield(address,address[]) (contracts/core/YieldManager.sol#246-325) has external calls inside a loop: processedWethByToken[address(token)] += weth.balanceOf(address(this)) - wethBalanceBefore (contracts/core/YieldManager.sol#311-313)
YieldManager.ethToRewards(uint256) (contracts/core/YieldManager.sol#427-463) has external calls inside a loop: tokenAddress = integrationMap.getTokenAddress(tokenId) (contracts/core/YieldManager.sol#441)
YieldManager.ethToRewards(uint256) (contracts/core/YieldManager.sol#427-463) has external calls inside a loop: IEtherRewards(moduleMap.getModuleAddress(Modules.EtherRewards)).increaseEthRewards(tokenAddress,(ethRewardsAmount * processedWethByToken[tokenAddress]) / processedWethByTokenSum) (contracts/core/YieldManager.sol#444-450)
YieldManager.getProcessedWethSum() (contracts/core/YieldManager.sol#557-573) has external calls inside a loop: tokenAddress = IIntegrationMap(moduleMap.getModuleAddress(Modules.IntegrationMap)).getTokenAddress(tokenId) (contracts/core/YieldManager.sol#568-570)
YieldManager.getTokenTotalIntegrationBalance(address) (contracts/core/YieldManager.sol#602-623) has external calls inside a loop: tokenTotalIntegrationBalance += IIntegration(integrationMap.getIntegrationAddress(integrationId)).getBalance(tokenAddress) (contracts/core/YieldManager.sol#619-621)
YieldManager.getAllPendingYield(address) (contracts/core/YieldManager.sol#690-720) has external calls inside a loop: integrationAddress = integrationMap.getIntegrationAddress(integrationId) (contracts/core/YieldManager.sol#708-710)
YieldManager.getAllPendingYield(address) (contracts/core/YieldManager.sol#690-720) has external calls inside a loop: pendingYield = IIntegration(integrationAddress).getPendingYield(token) (contracts/core/YieldManager.sol#711-712)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#calls-inside-a-loop

Reentrancy in YieldManager.distributeEth() (contracts/core/YieldManager.sol#328-369):
        External calls:
        - ethToGasAccount() (contracts/core/YieldManager.sol#336)
                - IWeth9(wethAddress).withdraw(ethAmountToGasAccount) (contracts/core/YieldManager.sol#391)
                - IWeth9(wethAddress).withdraw(ethAmountToGasAccount) (contracts/core/YieldManager.sol#398)
        - IERC20MetadataUpgradeable(wethAddress).safeTransfer(moduleMap.getModuleAddress(Modules.SushiSwapTrader),bioxBuyBackWethAmount) (contracts/core/YieldManager.sol#354-357)
        - IWeth9(wethAddress).withdraw(treasuryWethAmount) (contracts/core/YieldManager.sol#360)
        - ethToProtocolFeeAccrual(protocolFeeWethAmount) (contracts/core/YieldManager.sol#364)
                - IEtherRewards(moduleMap.getModuleAddress(Modules.EtherRewards)).increaseEthRewards(bioxAddress,protocolFeeEthRewardsAmount) (contracts/core/YieldManager.sol#481-482)
                - IWeth9(wethAddress).withdraw(protocolFeeEthRewardsAmount) (contracts/core/YieldManager.sol#484)
                - IERC20MetadataUpgradeable(wethAddress).transfer(moduleMap.getModuleAddress(Modules.Kernel),protocolFeeEthRewardsAmount) (contracts/core/YieldManager.sol#491-494)
        - ethToRewards(rewardsWethAmount) (contracts/core/YieldManager.sol#367)
                - IEtherRewards(moduleMap.getModuleAddress(Modules.EtherRewards)).increaseEthRewards(tokenAddress,(ethRewardsAmount * processedWethByToken[tokenAddress]) / processedWethByTokenSum) (contracts/core/YieldManager.sol#444-450)
                - IWeth9(wethAddress).withdraw(ethRewardsAmount) (contracts/core/YieldManager.sol#458)
        External calls sending eth:
        - ethToGasAccount() (contracts/core/YieldManager.sol#336)
                - gasAccount.transfer(ethAmountToGasAccount) (contracts/core/YieldManager.sol#392)
                - gasAccount.transfer(ethAmountToGasAccount) (contracts/core/YieldManager.sol#399)
        - address(treasuryAccount).transfer(treasuryWethAmount) (contracts/core/YieldManager.sol#361)
        - ethToProtocolFeeAccrual(protocolFeeWethAmount) (contracts/core/YieldManager.sol#364)
                - address(moduleMap.getModuleAddress(Modules.Kernel)).transfer(protocolFeeEthRewardsAmount) (contracts/core/YieldManager.sol#486-488)
        - ethToRewards(rewardsWethAmount) (contracts/core/YieldManager.sol#367)
                - address(moduleMap.getModuleAddress(Modules.Kernel)).transfer(ethRewardsAmount) (contracts/core/YieldManager.sol#460-462)
        State variables written after the call(s):
        - ethToRewards(rewardsWethAmount) (contracts/core/YieldManager.sol#367)
                - lastEthRewardsAmount = ethRewardsAmount (contracts/core/YieldManager.sol#456)
Reentrancy in YieldManager.harvestYield(address,address[]) (contracts/core/YieldManager.sol#246-325):
        External calls:
        - IIntegration(integrationAddress).harvestYield() (contracts/core/YieldManager.sol#262)
        State variables written after the call(s):
        - lastHarvestTimestampByIntegration[integrationAddress] = block.timestamp (contracts/core/YieldManager.sol#263)
        - processedWethByToken[address(weth)] += weth.balanceOf(address(this)) - wethBalanceBeforeHarvest (contracts/core/YieldManager.sol#265-267)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2

Reentrancy in YieldManager.harvestYield(address,address[]) (contracts/core/YieldManager.sol#246-325):
        External calls:
        - IIntegration(integrationAddress).harvestYield() (contracts/core/YieldManager.sol#262)
        Event emitted after the call(s):
        - HarvestYield(integrationAddress,tokenAddresses,harvestedWethAmounts) (contracts/core/YieldManager.sol#320-324)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3

AddressUpgradeable.isContract(address) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#26-35) uses assembly
        - INLINE ASM (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#33)
AddressUpgradeable._verifyCallResult(bool,bytes,string) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#147-164) uses assembly
        - INLINE ASM (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#156-159)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
```

AUTOMATED TESTING

```
Different versions of Solidity is used:
        - Version used: ['0.8.4', '>=0.7.6', '^0.8.0']
        - ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/proxy/utils/Initializable.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/IERC20Upgradeable.sol#3)
        - ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/extensions/IERC20MetadataUpgradeable.sol#3)
        - ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/utils/SafeERC20Upgradeable.sol#3)
        - ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#3)
        - 0.8.4 (contracts/core/Controlled.sol#2)
        - 0.8.4 (contracts/core/ModuleMapConsumer.sol#2)
        - 0.8.4 (contracts/core/YieldManager.sol#2)
        - 0.8.4 (contracts/interfaces/IABMIntegration.sol#2)
        - 0.8.4 (contracts/interfaces/IBiosRewards.sol#2)
        - 0.8.4 (contracts/interfaces/IEtherRewards.sol#2)
        - 0.8.4 (contracts/interfaces/IIntegration.sol#2)
        - 0.8.4 (contracts/interfaces/IIntegrationMap.sol#2)
        - 0.8.4 (contracts/interfaces/IKernel.sol#2)
        - 0.8.4 (contracts/interfaces/IModuleMap.sol#2)
        - 0.8.4 (contracts/interfaces/IStrategyMap.sol#2)
        - 0.8.4 (contracts/interfaces/ISushiSwapTrader.sol#2)
        - 0.8.4 (contracts/interfaces/ISwapManager.sol#1)
        - 0.8.4 (contracts/interfaces/IUserPositions.sol#2)
        - 0.8.4 (contracts/interfaces/IWeth9.sol#2)
        - 0.8.4 (contracts/interfaces/IYieldManager.sol#2)
        - >=0.7.6 (contracts/libraries/TokenSettings.sol#2)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used

AddressUpgradeable.functionCall(address,bytes) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#79-81) is never used and should be removed
AddressUpgradeable.functionCallWithValue(address,bytes,uint256) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#104-106) is never used and should be removed
AddressUpgradeable.functionStaticCall(address,bytes) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#129-131) is never used and should be removed
AddressUpgradeable.functionStaticCall(address,bytes,string) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#139-145) is never used and should be removed
AddressUpgradeable.sendValue(address,uint256) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#53-59) is never used and should be removed
SafeERC20Upgradeable.safeApprove(IERC20Upgradeable,address,uint256) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/utils/SafeERC20Upgradeable.sol#35-44) is never used and should be removed
SafeERC20Upgradeable.safeDecreaseAllowance(IERC20Upgradeable,address,uint256) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/utils/SafeERC20Upgradeable.sol#51-58) is never used and should be removed
SafeERC20Upgradeable.safeIncreaseAllowance(IERC20Upgradeable,address,uint256) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/utils/SafeERC20Upgradeable.sol#46-49) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/proxy/utils/Initializable.sol#4) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/IERC20Upgradeable.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/extensions/IERC20MetadataUpgradeable.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/utils/SafeERC20Upgradeable.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version0.8.4 (contracts/core/Controlled.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version0.8.4 (contracts/core/ModuleMapConsumer.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version0.8.4 (contracts/core/YieldManager.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version0.8.4 (contracts/interfaces/IABMIntegration.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version0.8.4 (contracts/interfaces/IBiosRewards.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version0.8.4 (contracts/interfaces/IEtherRewards.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version0.8.4 (contracts/interfaces/IIntegration.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version0.8.4 (contracts/interfaces/IIntegrationMap.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version0.8.4 (contracts/interfaces/IKernel.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version0.8.4 (contracts/interfaces/IModuleMap.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version0.8.4 (contracts/interfaces/IStrategyMap.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version0.8.4 (contracts/interfaces/ISushiSwapTrader.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version0.8.4 (contracts/interfaces/ISwapManager.sol#1) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version0.8.4 (contracts/interfaces/IUserPositions.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version0.8.4 (contracts/interfaces/IWeth9.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version0.8.4 (contracts/interfaces/IYieldManager.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
solc-0.8.4 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in AddressUpgradeable.sendValue(address,uint256) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#53-59):
        - (success) = recipient.call{value: amount}() (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#57)
Low level call in AddressUpgradeable.functionCallWithValue(address,bytes,uint256,string) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#114-121):
        - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#119)
Low level call in AddressUpgradeable.functionStaticCall(address,bytes,string) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#139-145):
        - (success,returndata) = target.staticcall(data) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#143)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Function Controlled.__Controlled_init(address[],address) (contracts/core/Controlled.sol#13-22) is not in mixedCase
Variable Controlled._controllers (contracts/core/Controlled.sol#10) is not in mixedCase
Function ModuleMapConsumer.__ModuleMapConsumer_init(address) (contracts/core/ModuleMapConsumer.sol#10-12) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Reentrancy in YieldManager.distributeEth() (contracts/core/YieldManager.sol#328-369):
        External calls:
        - ethToGasAccount() (contracts/core/YieldManager.sol#336)
                - gasAccount.transfer(ethAmountToGasAccount) (contracts/core/YieldManager.sol#392)
                - gasAccount.transfer(ethAmountToGasAccount) (contracts/core/YieldManager.sol#399)
        - address(treasuryAccount).transfer(treasuryWethAmount) (contracts/core/YieldManager.sol#361)
        - ethToProtocolFeeAccrual(protocolFeeWethAmount) (contracts/core/YieldManager.sol#364)
                - address(moduleMap.getModuleAddress(Modules.Kernel)).transfer(protocolFeeEthRewardsAmount) (contracts/core/YieldManager.sol#486-488)
        - ethToRewards(rewardsWethAmount) (contracts/core/YieldManager.sol#367)
                - address(moduleMap.getModuleAddress(Modules.Kernel)).transfer(ethRewardsAmount) (contracts/core/YieldManager.sol#460-462)
        State variables written after the call(s):
        - ethToRewards(rewardsWethAmount) (contracts/core/YieldManager.sol#367)
                - lastEthRewardsAmount = ethRewardsAmount (contracts/core/YieldManager.sol#456)
        - ethToRewards(rewardsWethAmount) (contracts/core/YieldManager.sol#367)
                - processedWethByToken[tokenAddress] = 0 (contracts/core/YieldManager.sol#452)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-4

Variable Controlled._controllers (contracts/core/Controlled.sol#10) is too similar to Controlled.__Controlled_init(address[],address).controllers_ (contracts/core/Controlled.sol#14)
Variable Controlled._controllers (contracts/core/Controlled.sol#10) is too similar to YieldManager.initialize(address[],address,uint256,uint32,uint32,uint32,uint32,address,address).controllers_ (contracts/core/YieldManager.sol#62)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar

initialize(address[],address,uint256,uint32,uint32,uint32,uint32,address,address) should be declared external:
        - YieldManager.initialize(address[],address,uint256,uint32,uint32,uint32,uint32,address,address) (contracts/core/YieldManager.sol#61-80)
harvestYield(address,address[]) should be declared external:
        - YieldManager.harvestYield(address,address[]) (contracts/core/YieldManager.sol#246-325)
getReserveTokenBalance(address) should be declared external:
        - YieldManager.getReserveTokenBalance(address) (contracts/core/YieldManager.sol#500-515)
getDesiredReserveTokenBalance(address) should be declared external:
        - YieldManager.getDesiredReserveTokenBalance(address) (contracts/core/YieldManager.sol#519-540)
getProcessedWethByToken(address) should be declared external:
        - YieldManager.getProcessedWethByToken(address) (contracts/core/YieldManager.sol#577-584)
getProcessedWethByTokenSum(address[]) should be declared external:
        - YieldManager.getProcessedWethByTokenSum(address[]) (contracts/core/YieldManager.sol#587-598)
getTokenTotalIntegrationBalance(address) should be declared external:
        - YieldManager.getTokenTotalIntegrationBalance(address) (contracts/core/YieldManager.sol#602-623)
getGasAccount() should be declared external:
        - YieldManager.getGasAccount() (contracts/core/YieldManager.sol#626-628)
getTreasuryAccount() should be declared external:
        - YieldManager.getTreasuryAccount() (contracts/core/YieldManager.sol#631-633)
getLastEthRewardsAmount() should be declared external:
        - YieldManager.getLastEthRewardsAmount() (contracts/core/YieldManager.sol#636-638)
getGasAccountTargetEthBalance() should be declared external:
        - YieldManager.getGasAccountTargetEthBalance() (contracts/core/YieldManager.sol#641-648)
getEthDistributionWeights() should be declared external:
        - YieldManager.getEthDistributionWeights() (contracts/core/YieldManager.sol#654-671)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

- No major issues were found by Slither.

THANK YOU FOR CHOOSING

// HALBORN