0xNodes - SushiSwap Integration

Smart Contract Security Audit

Prepared by: **Halborn**

Date of Engagement: **December 17th, 2021 - December 23rd, 2021**

Visit: **Halborn.com**

DOCUMENT REVISION HISTORY

VERSION	MODIFICATION	DATE	AUTHOR
0.1	Document Creation	12/17/2021	Roberto Reigada
0.2	Document Updates	12/23/2021	Roberto Reigada
0.3	Draft Review	12/23/2021	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 December 17th, 2021 and ending on December 23rd, 2021. 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 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.

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL

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

1.4 SCOPE

IN-SCOPE:

The security assessment was scoped to the following smart contracts:

- SushiSwapIntegrationV2.sol
- SushiSwapIntegration.sol
- UserPositions.sol

Commit ID: c8d81b2ea99a82df511d47a66c24121d131fd24d

EXECUTIVE OVERVIEW

2. ASSESSMENT SUMMARY & FINDINGS OVERVIEW

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL
0	0	2	2	6

LIKELIHOOD

IMPACT

🟨 (HAL-01) (HAL-02)	🟨	🟥	🟥	🟪
🟩	🟨	🟨	🟥	🟥
🟩	🟩	🟨	🟨	🟥
⬜	🟩 (HAL-03)	🟩 (HAL-04)	🟨	🟨
⬜ (HAL-05) (HAL-06) (HAL-07) (HAL-08) (HAL-09) (HAL-10)	⬜	🟩	🟩	🟨

SECURITY ANALYSIS	RISK LEVEL	REMEDIATION DATE
(HAL-01) INCOMPATIBILITY WITH NON-STANDARD ERC20 TOKENS	Medium	RISK ACCEPTED
(HAL-02) FRONT-RUNNING POSSIBILITY ON INITIALIZATION FUNCTIONS	Medium	RISK ACCPETED
(HAL-03) MISSING REQUIRE STATEMENT IN CREATEPOOL FUNCTION	Low	RISK ACCEPTED
(HAL-04) MISSING ZERO ADDRESS CHECKS	Low	RISK ACCEPTED
(HAL-05) SOLC 0.8.4 COMPILER VERSION CONTAINS MULTIPLE BUGS	Informational	ACKNOWLEDGED
(HAL-06) POSSIBLE MISUSE OF PUBLIC FUNCTIONS	Informational	ACKNOWLEDGED
(HAL-07) USING ++I CONSUMES LESS GAS THAN I++ IN LOOPS	Informational	ACKNOWLEDGED
(HAL-08) UNNEEDED INITIALIZATION OF UINT256 VARIABLES TO 0	Informational	ACKNOWLEDGED
(HAL-09) UINT32/UINT64 TYPES ARE LESS GAS EFFICIENT	Informational	ACKNOWLEDGED
(HAL-10) TYPO IN FUNCTION NAME	Informational	ACKNOWLEDGED

FINDINGS & TECH DETAILS

3.1 (HAL-01) INCOMPATIBILITY WITH NON-STANDARD ERC20 TOKENS - MEDIUM

Description:

In the contracts SushiSwapIntegrationV2, SushiSwapIntegration some functions perform a call to safeApprove().

Some tokens like USDT require that the allowance is zero before an approval call, for example:

Listing 1: USDT token approve function (Lines 205)

```
199 function approve(address _spender, uint _value) public onlyPayloadSize(2 * 32) {
200
201     // To change the approve amount you first have to reduce the addresses`
202     //  allowance to zero by calling `approve(_spender, 0)` if it is not
203     //  already 0 to mitigate the race condition described here:
204     //  https://github.com/ethereum/EIPs/issues/20#issuecomment-263524729
205     require(!((_value != 0) && (allowed[msg.sender][_spender] != 0)));
206
207     allowed[msg.sender][_spender] = _value;
208     Approval(msg.sender, _spender, _value);
209 }
```

This is not considered by SushiSwapIntegrationV2, SushiSwapIntegration. With the current implementation, if for example USDT was used as the tokenIn the call to swapExactInput() would revert.

Code Location:

SushiSwapIntegrationV2

- Line 164:

```
IERC20MetadataUpgradeable(pairAddress).safeApprove(masterChef,type(
uint256).max);
```

- Line 548:

```
IERC20MetadataUpgradeable(pairFor(pool.tokenA, pool.tokenB)).
safeApprove(swapRouterAddress, liquidityToWithdraw);
```

- Line 625:

```
IERC20MetadataUpgradeable(tokenIn).safeApprove(swapRouterAddress,
amountIn);
```

SushiSwapIntegration

- Line 102:

```
IERC20MetadataUpgradeable(tokenA).safeApprove(swapRouterAddress, type(
uint256).max);
```

- Line 114:

```
IERC20MetadataUpgradeable(tokenB).safeApprove(swapRouterAddress, type(
uint256).max);
```

- Line 167:

```
IERC20MetadataUpgradeable(pairAddress).safeApprove(masterChef, type(
uint256).max);
```

- Line 489:

```
IERC20MetadataUpgradeable(pairFor(pool.tokenA, pool.tokenB)).
safeApprove(swapRouterAddress, liquidityToWithdraw);
```

- Line 583:

```
IERC20MetadataUpgradeable(tokenIn).safeApprove(swapRouterAddress,
amountIn);
```

Risk Level:

Likelihood - 1
Impact - 5

Recommendation:

It is recommended calling `safeApprove(<address>, 0)` before calling `safeApprove(<address>, <amount>)`.

Remediation Plan:

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

3.2 (HAL-02) FRONT-RUNNING POSSIBILITY ON INITIALIZATION FUNCTIONS - MEDIUM

Description:

The contracts SushiSwapIntegrationV2, SushiSwapIntegration and UserPositions have initialization functions that can be front-run. These smart contracts should be deployed using a factory pattern to prevent the front-running:
https://docs.openzeppelin.com/learn/upgrading-smart-contracts

As we can see, the deployment scripts given with the project do not make use of any factory pattern:

Listing 2: 008_deploy_sushiswap_integration_V2.ts (Lines 17)

```
16 console.log("deploying SushiSwapIntegrationV2...");
17 const sushiIntegrationV2 = await deployContract(
18   hre,
19   "SushiSwapIntegrationV2",
20   [
21     [(await hre.ethers.getContract("YieldManager")).address],
22     (await hre.ethers.getContract("ModuleMap")).address,
23     addresses.sushiSwapFactoryAddress,
24     addresses.sushiSwapRouterAddress,
25     addresses.masterChefV2,
26     addresses.sushi,
27     slippageNumerator,
28   ]
29 );
```

Listing 3: 007_deploy_sushiswap_integration.ts (Lines 17)

```
15 console.log("Deploying SushiSwapV1...");
16
17 const sushiIntegration = await deployContract(hre, "
      SushiSwapIntegration", [
18   [(await hre.ethers.getContract("YieldManager")).address],
19   (await hre.ethers.getContract("ModuleMap")).address,
20   addresses.sushiSwapFactoryAddress,
21   addresses.sushiSwapRouterAddress,
22   addresses.masterChef,
23   addresses.sushi,
24   slippageNumerator,
25 ]);
```

Listing 4: 003_deploy_user_positions.ts (Lines 7)

```
 5 const func: DeployFunction = async (hre: HardhatRuntimeEnvironment
     ) => {
 6   const defaultBiosRewardsDuration = 2592000;
 7   const userPositionsResult = await deployContract(hre, "
       UserPositions", [
 8     [(await hre.ethers.getContract("Kernel")).address],
 9     (await hre.ethers.getContract("ModuleMap")).address,
10     defaultBiosRewardsDuration,
11   ]);
12   await setModuleAddress(hre, 1, userPositionsResult.address);
13 };
```

Risk Level:

Likelihood - 1
Impact - 5

Recommendation:

It is recommended to use a factory pattern that will deploy and initialize the contracts atomically to prevent front-running of the initialization.

Remediation Plan:

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

3.3 (HAL-03) MISSING REQUIRE STATEMENT IN UPDATESTAKELOCK FUNCTION - LOW

Description:

The contracts SushiSwapIntegrationV2 and SushiSwapIntegration contain the function createPool(). This function is missing a require statement that checks that the tokenA and tokenB are not the same:

Listing 5: SushiSwapIntegrationV2.sol

```
function createPool(
    address tokenA,
    address tokenB,
    uint256
) external override onlyManager {
    poolCount++;
    pools[poolCount].tokenA = tokenA;
    pools[poolCount].tokenB = tokenB;
    poolIds.push(poolCount);

    if (
        IERC20MetadataUpgradeable(tokenA).allowance(
            address(this),
            swapRouterAddress
        ) == 0
    ) {
        IERC20MetadataUpgradeable(tokenA).safeApprove(
            swapRouterAddress,
            type(uint256).max
        );
    }

    if (
        IERC20MetadataUpgradeable(tokenB).allowance(
            address(this),
            swapRouterAddress
        ) == 0
    ) {
        IERC20MetadataUpgradeable(tokenB).safeApprove(
            swapRouterAddress,
            type(uint256).max
        );
    }
}
```

Risk Level:

Likelihood - 2
Impact - 2

Recommendation:

It is recommended to add a require statement in the createPool() function that checks that the tokenA and tokenB are not the same.

Remediation Plan:

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

3.4 (HAL-04) MISSING ZERO ADDRESS CHECKS - LOW

Description:

The contracts SushiSwapIntegrationV2, SushiSwapIntegration and UserPositions are missing address validation in their initialize functions. Every address should be validated and checked that is different from zero.

Code location:

Listing 6: SushiSwapIntegrationV2.sol

```
function initialize(
    address[] memory controllers_,
    address moduleMap_,
    address factoryAddress_,
    address swapRouterAddress_,
    address masterChef_,
    address sushi_,
    uint24 slippageNumerator_
) public initializer {
    __Controlled_init(controllers_, moduleMap_);
    factoryAddress = factoryAddress_;
    swapRouterAddress = swapRouterAddress_;
    masterChef = masterChef_;
    slippageNumerator = slippageNumerator_;
    sushi = sushi_;
    wethAddress = IIntegrationMap(
        moduleMap.getModuleAddress(Modules.IntegrationMap)
    ).getWethTokenAddress();
}
```

Listing 7: SushiSwapIntegration.sol

```
function initialize(
    address[] memory controllers_,
    address moduleMap_,
    address factoryAddress_,
    address swapRouterAddress_,
    address masterChef_,
    address sushi_,
    uint24 slippageNumerator_
) public initializer {
    __Controlled_init(controllers_, moduleMap_);
    factoryAddress = factoryAddress_;
    swapRouterAddress = swapRouterAddress_;
    masterChef = masterChef_;
    slippageNumerator = slippageNumerator_;
    sushi = sushi_;
    wethAddress = IIntegrationMap(
        moduleMap.getModuleAddress(Modules.IntegrationMap)
    ).getWethTokenAddress();
}
```

Listing 8: UserPositions.sol

```
function initialize(
    address[] memory controllers_,
    address moduleMap_,
    uint32 biosRewardsDuration_
) public initializer {
    __Controlled_init(controllers_, moduleMap_);
    _biosRewardsDuration = biosRewardsDuration_;
}
```

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.

3.5 (HAL-05) SOLC 0.8.4 COMPILER VERSION CONTAINS MULTIPLE BUGS - INFORMATIONAL

Description:

Solidity compiler version 0.8.3, 0.8.4 and 0.8.9 fixed important bugs in the compiler. The version 0.8.4 is missing the following 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 ^0.8.0 is still chosen to be used because of the new functionality it provides, it is recommended to use the 0.8.10 version.

Remediation Plan:

ACKNOWLEDGED: The 0xNodes team acknowledged this issue.

3.6 (HAL-06) 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:

SushiSwapIntegrationV2.sol
- initialize() (SushiSwapIntegrationV2.sol#65-83)
- withdraw() (SushiSwapIntegrationV2.sol#409-423)
- getTokensOrdered() (SushiSwapIntegrationV2.sol#729-741)

SushiSwapIntegration.sol
- initialize() (SushiSwapIntegration.sol#65-83)
- withdraw() (SushiSwapIntegration.sol#351-365)

UserPositions.sol
- initialize() (UserPositions.sol#47-54)
- totalTokenBalance() (UserPositions.sol#460-467)
- getBiosRewardsDuration() (UserPositions.sol#484-486)

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.

3.7 (HAL-07) USING ++I CONSUMES LESS GAS THAN I++ IN LOOPS - INFORMATIONAL

Description:

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

Code Location:

SushiSwapIntegrationV2.sol
- Line 307: `for (uint32 i = 0; i < poolCount; i++){`

SushiSwapIntegration.sol
- Line 306: `for (uint32 i = 0; i < poolCount; i++){`

UserPositions.sol
- Line 118: `for (uint256 tokenId; tokenId < tokens.length; tokenId++){`
- Line 223: `for (uint256 tokenId; tokenId < tokens.length; tokenId++){`
- Line 268: `for (uint256 tokenId; tokenId < tokens.length; tokenId++){`
- Line 362: `for (uint256 tokenId; tokenId < tokenCount; tokenId++){`
- Line 440: `for (uint256 tokenId; tokenId < tokenCount; tokenId++){`
- Line 502: `for (uint256 i = 0; i < tokens.length; i++){`
- Line 558: `for (uint256 i = 0; i < _tokens.length; i++){`
- Line 565: `for (uint256 i = 0; i < _strategies.length; i++){`
- Line 574: `for (uint256 j = 0; j < strategyTokens.length; j++){`
- Line 591: `for (uint256 i = 0; i < tokens.length; i++){`
- Line 620: `for (uint256 i = 0; i < users.length; i++){`

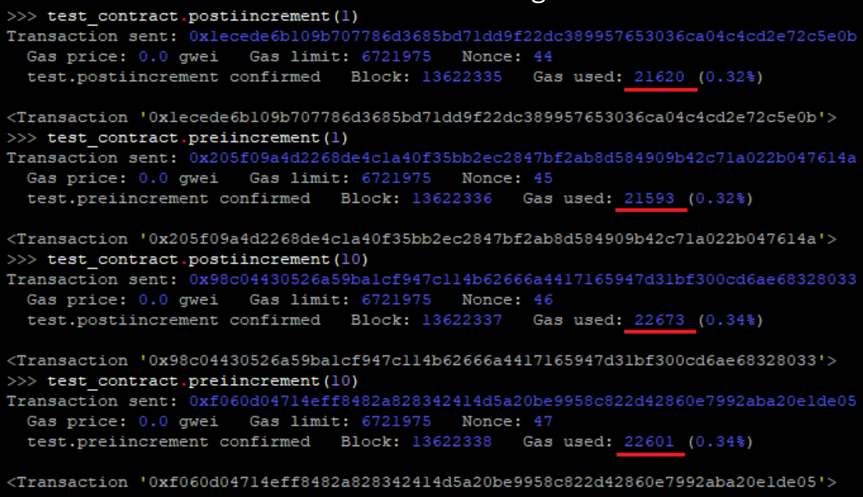
Proof of Concept:

For example, based in the following test contract:

Listing 9: Test.sol

```
//SPDX-License-Identifier: MIT
pragma solidity 0.8.9;

contract test {
    function postiincrement(uint256 iterations) public {
        for (uint256 i = 0; i < iterations; i++) {
        }
    }
    function preiincrement(uint256 iterations) public {
        for (uint256 i = 0; i < iterations; ++i) {
        }
    }
}
```

We can see the difference in the gas costs:



```
>>> test_contract.postiincrement(1)
Transaction sent: 0x1ecede6b109b707786d3685bd71dd9f22dc389957653036ca04c4cd2e72c5e0b
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 44
  test.postiincrement confirmed   Block: 13622335   Gas used: 21620 (0.32%)

<Transaction '0x1ecede6b109b707786d3685bd71dd9f22dc389957653036ca04c4cd2e72c5e0b'>
>>> 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: 0x98c04430526a59ba1cf947c114b62666a4417165947d31bf300cd6ae68328033
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 46
  test.postiincrement confirmed   Block: 13622337   Gas used: 22673 (0.34%)

<Transaction '0x98c04430526a59ba1cf947c114b62666a4417165947d31bf300cd6ae68328033'>
>>> test_contract.preiincrement(10)
Transaction sent: 0xf060d04714eff8482a828342414d5a20be9958c822d42860e7992aba20e1de05
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 47
  test.preiincrement confirmed   Block: 13622338   Gas used: 22601 (0.34%)

<Transaction '0xf060d04714eff8482a828342414d5a20be9958c822d42860e7992aba20e1de05'>
```

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.

3.8 (HAL-08) UNNEEDED INITIALIZATION OF UINT256 VARIABLES TO 0 - INFORMATIONAL

Description:

As `i` is an `uint`, it is already initialized to 0. `uint i = 0` reassigns the 0 to `i` which wastes gas.

Code Location:

SushiSwapIntegrationV2.sol
- Line 307: `for (uint32 i = 0; i < poolCount; i++){`

SushiSwapIntegration.sol
- Line 306: `for (uint32 i = 0; i < poolCount; i++){`

UserPositions.sol
- Line 502: `for (uint256 i = 0; i < tokens.length; i++){`
- Line 558: `for (uint256 i = 0; i < _tokens.length; i++){`
- Line 565: `for (uint256 i = 0; i < _strategies.length; i++){`
- Line 574: `for (uint256 j = 0; j < strategyTokens.length; j++){`
- Line 591: `for (uint256 i = 0; i < tokens.length; i++){`
- Line 620: `for (uint256 i = 0; i < users.length; i++){`

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

It is recommended to not initialize `i` variable to 0 to save some gas. For example:

```
for (uint256 i; i < tokens.length; ++i){
```

Remediation Plan:

ACKNOWLEDGED: The 0xNodes team acknowledged this issue.

3.9 (HAL-09) 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 numbers introduces additional costs as the EVM has to properly enforce the limits of this smaller type.

This issue happens with multiple functions and state variables of the smart contracts.

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.

Code Location:

SushiSwapIntegrationV2.sol

- Line 22: uint24 private constant SLIPPAGE_DENOMINATOR = 1_000_000;
- Line 23: uint24 slippageNumerator;
- Line 32: uint32 public poolCount;
- Line 33: uint32[] private poolIds;
- Line 35: mapping(uint32 => uint256)private stakings;
- Line 45: mapping(uint32 => uint256)public yieldBalances;
- Line 72: uint24 slippageNumerator_
- Line 121: function configureStaking(uint32 poolId, uint256 masterChefPoolId)
- Line 128: function configureStackingOverwrite(uint32 poolId, uint256 masterChefPoolId)
- Line 136: uint32 poolId,
- Line 176: uint32 poolId
- Line 184: uint32 poolId,
- Line 191: function getPool(uint32 pid)public view override returns (Pool memory){
- Line 196: function getBalance(uint32 poolId, address token)

- Line 204: function getPoolBalance(uint32 poolId)
- Line 213: function deploy(uint32 poolId)external override onlyController {
- Line 217: function manualDeploy(uint32 poolId)external override onlyManager {
- Line 221: function _deploy(uint32 poolId)internal {
- Line 291: function stakeLPTokens(uint32 poolId)external onlyManager {
- Line 307: for (uint32 i = 0; i < poolCount; i++){
- Line 321: uint32 poolId
- Line 412: uint32 poolId
- Line 425: function getTokensPoolValue(uint32 poolId)
- Line 455: uint32 poolId,
- Line 503: uint32 poolId,
- Line 520: uint32 poolId,
- Line 764: uint32 poolId,

SushiSwapIntegration.sol

- Line 22: uint24 private constant SLIPPAGE_DENOMINATOR = 1_000_000;
- Line 23: uint24 slippageNumerator;
- Line 32: uint32 public poolCount;
- Line 33: uint32[] private poolIds;
- Line 35: mapping(uint32 => uint256)private stakings;
- Line 45: mapping(uint32 => uint256)public yieldBalances;
- Line 72: uint24 slippageNumerator_
- Line 121: function configureStaking(uint32 poolId, uint256 masterChefPoolId)
- Line 128: function configureStakingOverwrite(uint32 poolId, uint256 masterChefPoolId)
- Line 136: uint32 poolId,
- Line 179: uint32 poolId
- Line 187: uint32 poolId,
- Line 194: function getPool(uint32 pid)public view override returns (Pool memory){
- Line 199: function getBalance(uint32 poolId, address token)
- Line 207: function getPoolBalance(uint32 poolId)
- Line 216: function deploy(uint32 poolId)external override onlyController {
- Line 220:

function manualDeploy(uint32 poolId)external override onlyManager {
- Line 224: function _deploy(uint32 poolId)internal {
- Line 294: function stakeLPTokens(uint32 poolId)external onlyManager {
- Line 306: for (uint32 i = 0; i < poolCount; i++){
- Line 320: uint32 poolId
- Line 354: uint32 poolId
- Line 370: uint32 poolId
- Line 384: function getTokensPoolValue(uint32 poolId)
- Line 414: uint32 poolId,
- Line 462: uint32 poolId,
- Line 515: uint32 poolId,
- Line 707: uint32 poolId,

UserPositions.sol
- Line 29: uint32 private _biosRewardsDuration;
- Line 50: uint32 biosRewardsDuration_
- Line 57: function setBiosRewardsDuration(uint32 biosRewardsDuration_)
- Line 484:

function getBiosRewardsDuration()public view override returns (uint32){

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

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

Remediation Plan:

ACKNOWLEDGED: The 0xNodes team acknowledged this issue.

3.10 (HAL-10) TYPO IN FUNCTION NAME - INFORMATIONAL

Description:

In the contract SushiSwapIntegrationV2 there is a function called configureStackingOverwrite when it actually should be called configureStakingOverwrite, as it is done in the contract SushiSwapIntegration.

Code Location:

Listing 10: SushiSwapIntegrationV2.sol (Lines 128)

```
128 function configureStackingOverwrite(uint32 poolId, uint256
        masterChefPoolId)
129      external
130      onlyManager
131 {
132      _configureStaking(poolId, masterChefPoolId, true);
133 }
```

Listing 11: SushiSwapIntegration.sol (Lines 128)

```
128 function configureStakingOverwrite(uint32 poolId, uint256
        masterChefPoolId)
129      external
130      onlyManager
131 {
132      _configureStaking(poolId, masterChefPoolId, true);
133 }
```

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

It is recommended to rename the function SushiSwapIntegrationV2.configureStackingOverwrite() to configureStakingOverwrite.

Remediation Plan:

ACKNOWLEDGED: The 0xNodes team acknowledged this issue.

AUTOMATED TESTING

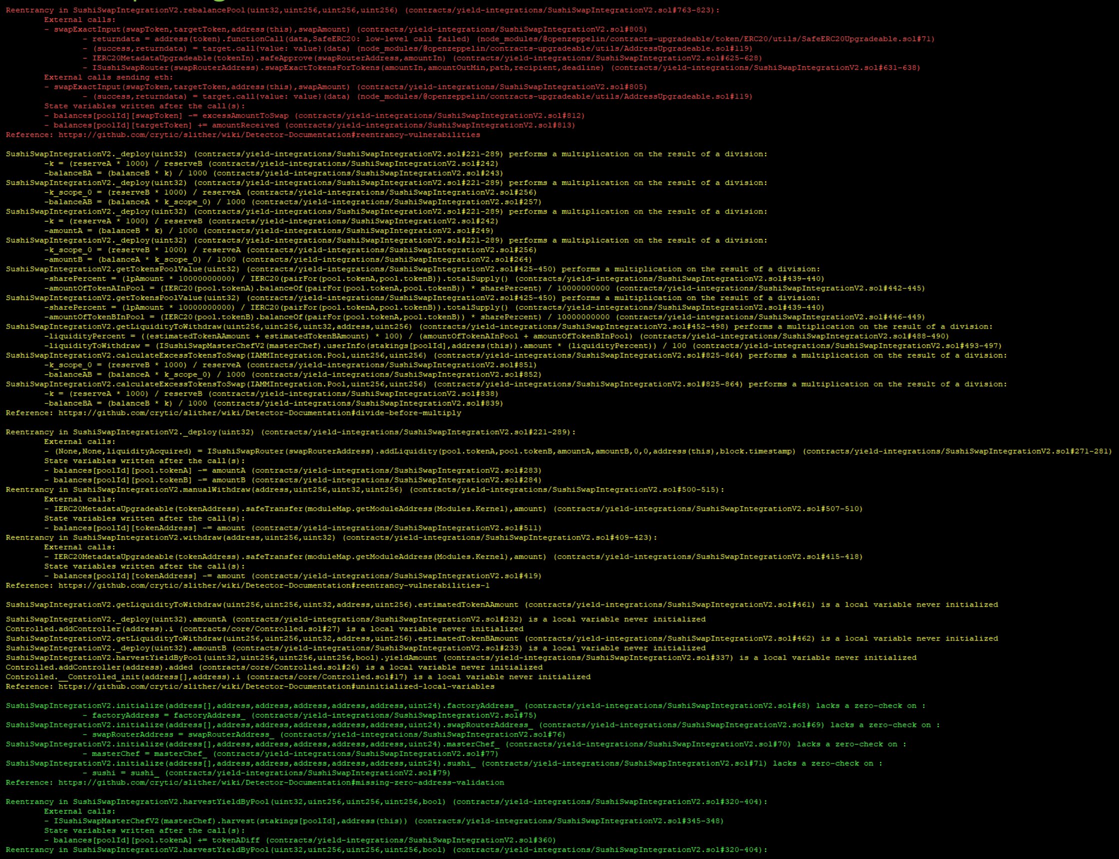
4.1 STATIC ANALYSIS REPORT

Description:

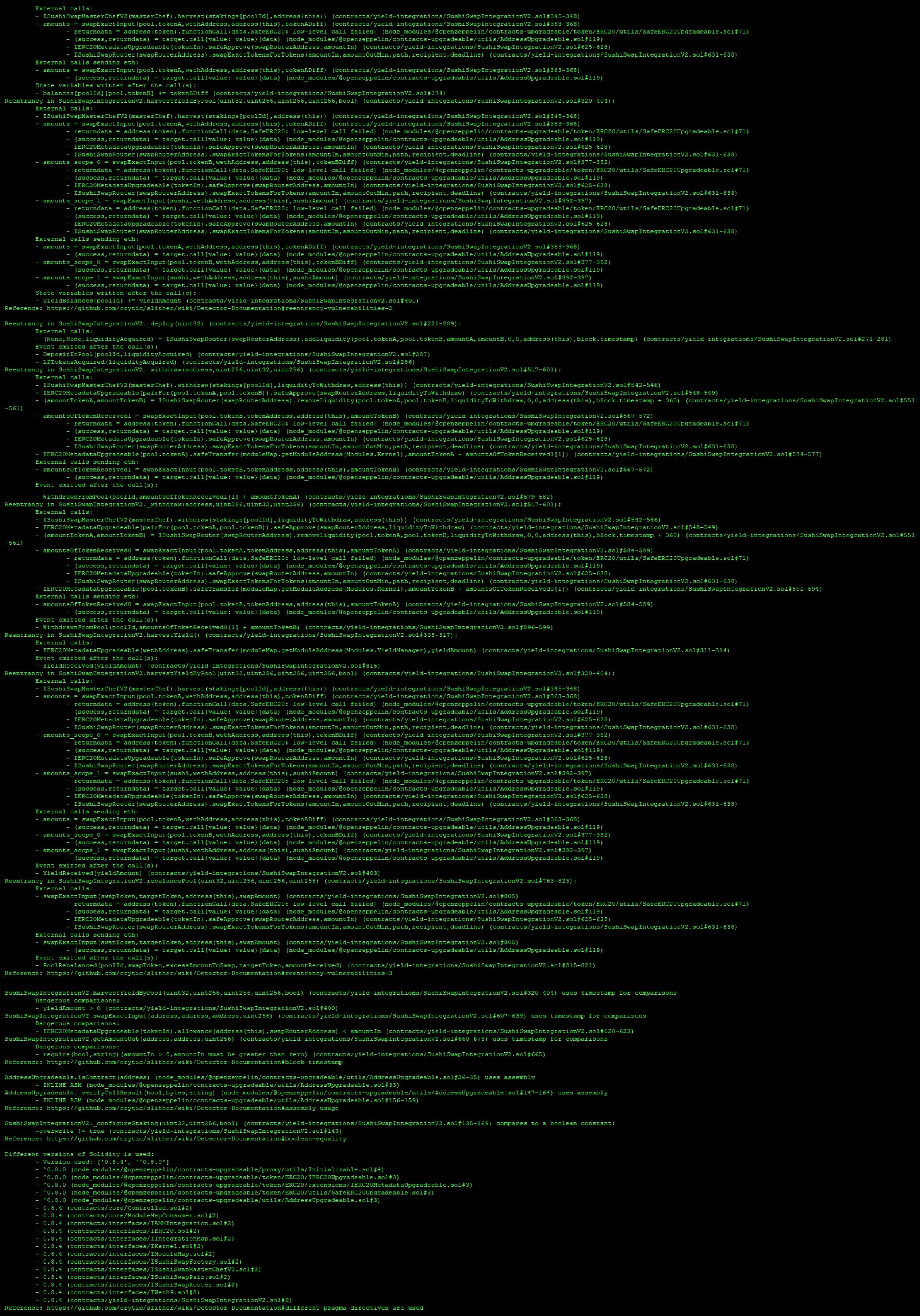
Halborn used automated testing techniques to enhance the coverage of certain areas of the scoped contracts. Among the tools used was Slither, a Solidity static analysis framework. After Halborn verified all the contracts in the repository and was able to compile them correctly into their abi and binary 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:

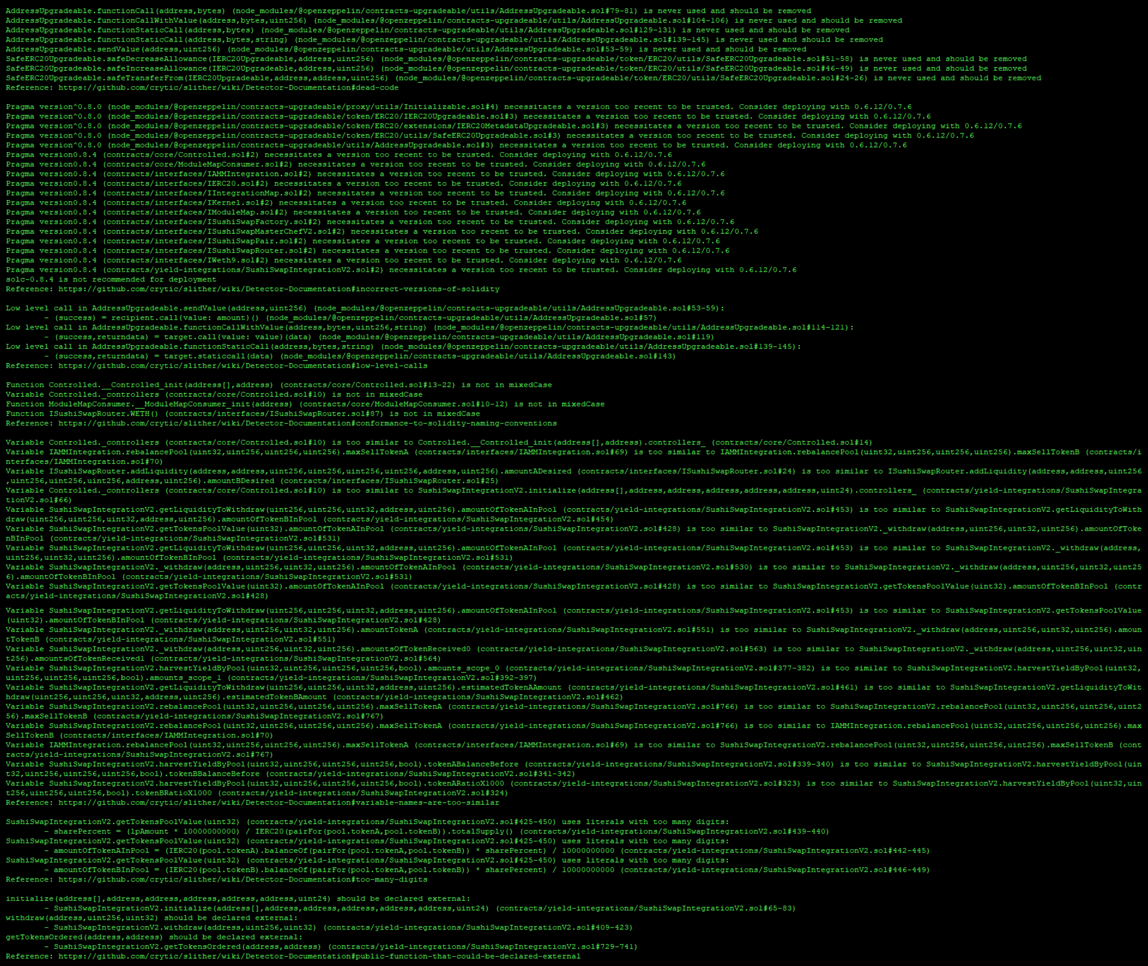
SushiSwapIntegrationV2.sol



```
Reentrancy in SushiSwapIntegrationV2.rebalancePool(uint32,uint256,uint256,uint256) (contracts/yield-integrations/SushiSwapIntegrationV2.sol#763-823):
        External calls:
        - swapExactInput(swapToken,targetToken,address(this),swapAmount) (contracts/yield-integrations/SushiSwapIntegrationV2.sol#805)
                - returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/utils/SafeERC20Upgradeable.sol#71)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#119)
                - IERC20MetadataUpgradeable(tokenIn).safeApprove(swapRouterAddress,amountIn) (contracts/yield-integrations/SushiSwapIntegrationV2.sol#625-628)
                - ISushiSwapRouter(swapRouterAddress).swapExactTokensForTokens(amountIn,amountOutMin,path,recipient,deadline) (contracts/yield-integrations/SushiSwapIntegrationV2.sol#631-638)
        External calls sending eth:
        - swapExactInput(swapToken,targetToken,address(this),swapAmount) (contracts/yield-integrations/SushiSwapIntegrationV2.sol#805)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#119)
        State variables written after the call(s):
        - balances[poolId][swapToken] -= excessAmountToSwap (contracts/yield-integrations/SushiSwapIntegrationV2.sol#812)
        - balances[poolId][targetToken] += amountReceived (contracts/yield-integrations/SushiSwapIntegrationV2.sol#813)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities

SushiSwapIntegrationV2._deploy(uint32) (contracts/yield-integrations/SushiSwapIntegrationV2.sol#221-289) performs a multiplication on the result of a division:
        -k = (reserveA * 1000) / reserveB (contracts/yield-integrations/SushiSwapIntegrationV2.sol#242)
        -balanceBA = (balanceB * k) / 1000 (contracts/yield-integrations/SushiSwapIntegrationV2.sol#243)
SushiSwapIntegrationV2._deploy(uint32) (contracts/yield-integrations/SushiSwapIntegrationV2.sol#221-289) performs a multiplication on the result of a division:
        -k_scope_0 = (reserveB * 1000) / reserveA (contracts/yield-integrations/SushiSwapIntegrationV2.sol#256)
        -balanceAB = (balanceA * k_scope_0) / 1000 (contracts/yield-integrations/SushiSwapIntegrationV2.sol#257)
SushiSwapIntegrationV2._deploy(uint32) (contracts/yield-integrations/SushiSwapIntegrationV2.sol#221-289) performs a multiplication on the result of a division:
        -k = (reserveA * 1000) / reserveB (contracts/yield-integrations/SushiSwapIntegrationV2.sol#242)
        -amountA = (balanceB * k) / 1000 (contracts/yield-integrations/SushiSwapIntegrationV2.sol#249)
SushiSwapIntegrationV2._deploy(uint32) (contracts/yield-integrations/SushiSwapIntegrationV2.sol#221-289) performs a multiplication on the result of a division:
        -k_scope_0 = (reserveB * 1000) / reserveA (contracts/yield-integrations/SushiSwapIntegrationV2.sol#256)
        -amountB = (balanceA * k_scope_0) / 1000 (contracts/yield-integrations/SushiSwapIntegrationV2.sol#264)
SushiSwapIntegrationV2.getTokensPoolValue(uint32) (contracts/yield-integrations/SushiSwapIntegrationV2.sol#425-450) performs a multiplication on the result of a division:
        -sharePercent = (lpAmount * 10000000000) / IERC20(pairFor(pool.tokenA,pool.tokenB)).totalSupply() (contracts/yield-integrations/SushiSwapIntegrationV2.sol#439-440)
        -amountOfTokenAInPool = (IERC20(pool.tokenA).balanceOf(pairFor(pool.tokenA,pool.tokenB)) * sharePercent) / 10000000000 (contracts/yield-integrations/SushiSwapIntegrationV2.sol#442-445)
SushiSwapIntegrationV2.getTokensPoolValue(uint32) (contracts/yield-integrations/SushiSwapIntegrationV2.sol#425-450) performs a multiplication on the result of a division:
        -sharePercent = (lpAmount * 10000000000) / IERC20(pairFor(pool.tokenA,pool.tokenB)).totalSupply() (contracts/yield-integrations/SushiSwapIntegrationV2.sol#439-440)
        -amountOfTokenBInPool = (IERC20(pool.tokenB).balanceOf(pairFor(pool.tokenA,pool.tokenB)) * sharePercent) / 10000000000 (contracts/yield-integrations/SushiSwapIntegrationV2.sol#446-449)
SushiSwapIntegrationV2.getLiquidityToWithdraw(uint256,uint256,uint32,address,uint256) (contracts/yield-integrations/SushiSwapIntegrationV2.sol#452-498) performs a multiplication on the result of a division:
        -liquidityPercent = ((estimatedTokenAAmount + estimatedTokenBAmount) * 100) / (amountOfTokenAInPool + amountOfTokenBInPool) (contracts/yield-integrations/SushiSwapIntegrationV2.sol#488-490)
        -liquidityToWithdraw = (ISushiSwapMasterChefV2(masterChef).userInfo(stakings[poolId],address(this)).amount * (liquidityPercent)) / 100 (contracts/yield-integrations/SushiSwapIntegrationV2.sol#493-497)
SushiSwapIntegrationV2.calculateExcessTokensToSwap(IAMMIntegration.Pool,uint256,uint256) (contracts/yield-integrations/SushiSwapIntegrationV2.sol#825-864) performs a multiplication on the result of a division:
        -k_scope_0 = (reserveB * 1000) / reserveA (contracts/yield-integrations/SushiSwapIntegrationV2.sol#851)
        -balanceAB = (balanceA * k_scope_0) / 1000 (contracts/yield-integrations/SushiSwapIntegrationV2.sol#852)
SushiSwapIntegrationV2.calculateExcessTokensToSwap(IAMMIntegration.Pool,uint256,uint256) (contracts/yield-integrations/SushiSwapIntegrationV2.sol#825-864) performs a multiplication on the result of a division:
        -k = (reserveA * 1000) / reserveB (contracts/yield-integrations/SushiSwapIntegrationV2.sol#838)
        -balanceBA = (balanceB * k) / 1000 (contracts/yield-integrations/SushiSwapIntegrationV2.sol#839)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply

Reentrancy in SushiSwapIntegrationV2._deploy(uint32) (contracts/yield-integrations/SushiSwapIntegrationV2.sol#221-289):
        External calls:
        - (None,None,liquidityAcquired) = ISushiSwapRouter(swapRouterAddress).addLiquidity(pool.tokenA,pool.tokenB,amountA,amountB,0,0,address(this),block.timestamp) (contracts/yield-integrations/SushiSwapIntegrationV2.sol#271-281)
        State variables written after the call(s):
        - balances[poolId][pool.tokenA] -= amountA (contracts/yield-integrations/SushiSwapIntegrationV2.sol#283)
        - balances[poolId][pool.tokenB] -= amountB (contracts/yield-integrations/SushiSwapIntegrationV2.sol#284)
Reentrancy in SushiSwapIntegrationV2.manualWithdraw(address,uint256,uint32,uint256) (contracts/yield-integrations/SushiSwapIntegrationV2.sol#500-515):
        External calls:
        - IERC20MetadataUpgradeable(tokenAddress).safeTransfer(moduleMap.getModuleAddress(Modules.Kernel),amount) (contracts/yield-integrations/SushiSwapIntegrationV2.sol#507-510)
        State variables written after the call(s):
        - balances[poolId][tokenAddress] -= amount (contracts/yield-integrations/SushiSwapIntegrationV2.sol#511)
Reentrancy in SushiSwapIntegrationV2.withdraw(address,uint256,uint32) (contracts/yield-integrations/SushiSwapIntegrationV2.sol#409-423):
        External calls:
        - IERC20MetadataUpgradeable(tokenAddress).safeTransfer(moduleMap.getModuleAddress(Modules.Kernel),amount) (contracts/yield-integrations/SushiSwapIntegrationV2.sol#415-418)
        State variables written after the call(s):
        - balances[poolId][tokenAddress] -= amount (contracts/yield-integrations/SushiSwapIntegrationV2.sol#419)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1

SushiSwapIntegrationV2.getLiquidityToWithdraw(uint256,uint256,uint32,address,uint256).estimatedTokenAAmount (contracts/yield-integrations/SushiSwapIntegrationV2.sol#461) is a local variable never initialized

SushiSwapIntegrationV2._deploy(uint32).amountA (contracts/yield-integrations/SushiSwapIntegrationV2.sol#232) is a local variable never initialized
Controlled.addController(address).i (contracts/core/Controlled.sol#27) is a local variable never initialized
SushiSwapIntegrationV2.getLiquidityToWithdraw(uint256,uint256,uint32,address,uint256).estimatedTokenBAmount (contracts/yield-integrations/SushiSwapIntegrationV2.sol#462) is a local variable never initialized
SushiSwapIntegrationV2._deploy(uint32).amountB (contracts/yield-integrations/SushiSwapIntegrationV2.sol#233) is a local variable never initialized
SushiSwapIntegrationV2.harvestYieldByPool(uint32,uint256,uint256,uint256,bool).yieldAmount (contracts/yield-integrations/SushiSwapIntegrationV2.sol#337) is a local variable never initialized
Controlled.addController(address).added (contracts/core/Controlled.sol#26) is a local variable never initialized
Controlled.__Controlled_init(address[],address).i (contracts/core/Controlled.sol#17) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables

SushiSwapIntegrationV2.initialize(address[],address,address,address,address,address,uint24).factoryAddress_ (contracts/yield-integrations/SushiSwapIntegrationV2.sol#68) lacks a zero-check on :
                - factoryAddress = factoryAddress_ (contracts/yield-integrations/SushiSwapIntegrationV2.sol#75)
SushiSwapIntegrationV2.initialize(address[],address,address,address,address,address,uint24).swapRouterAddress_ (contracts/yield-integrations/SushiSwapIntegrationV2.sol#69) lacks a zero-check on :
                - swapRouterAddress = swapRouterAddress_ (contracts/yield-integrations/SushiSwapIntegrationV2.sol#76)
SushiSwapIntegrationV2.initialize(address[],address,address,address,address,address,uint24).masterChef_ (contracts/yield-integrations/SushiSwapIntegrationV2.sol#70) lacks a zero-check on :
                - masterChef = masterChef_ (contracts/yield-integrations/SushiSwapIntegrationV2.sol#77)
SushiSwapIntegrationV2.initialize(address[],address,address,address,address,address,uint24).sushi_ (contracts/yield-integrations/SushiSwapIntegrationV2.sol#71) lacks a zero-check on :
                - sushi = sushi_ (contracts/yield-integrations/SushiSwapIntegrationV2.sol#79)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

Reentrancy in SushiSwapIntegrationV2.harvestYieldByPool(uint32,uint256,uint256,uint256,bool) (contracts/yield-integrations/SushiSwapIntegrationV2.sol#320-404):
        External calls:
        - ISushiSwapMasterChefV2(masterChef).harvest(stakings[poolId],address(this)) (contracts/yield-integrations/SushiSwapIntegrationV2.sol#345-348)
        State variables written after the call(s):
        - balances[poolId][pool.tokenA] += tokenADiff (contracts/yield-integrations/SushiSwapIntegrationV2.sol#360)
Reentrancy in SushiSwapIntegrationV2.harvestYieldByPool(uint32,uint256,uint256,uint256,bool) (contracts/yield-integrations/SushiSwapIntegrationV2.sol#320-404):
```

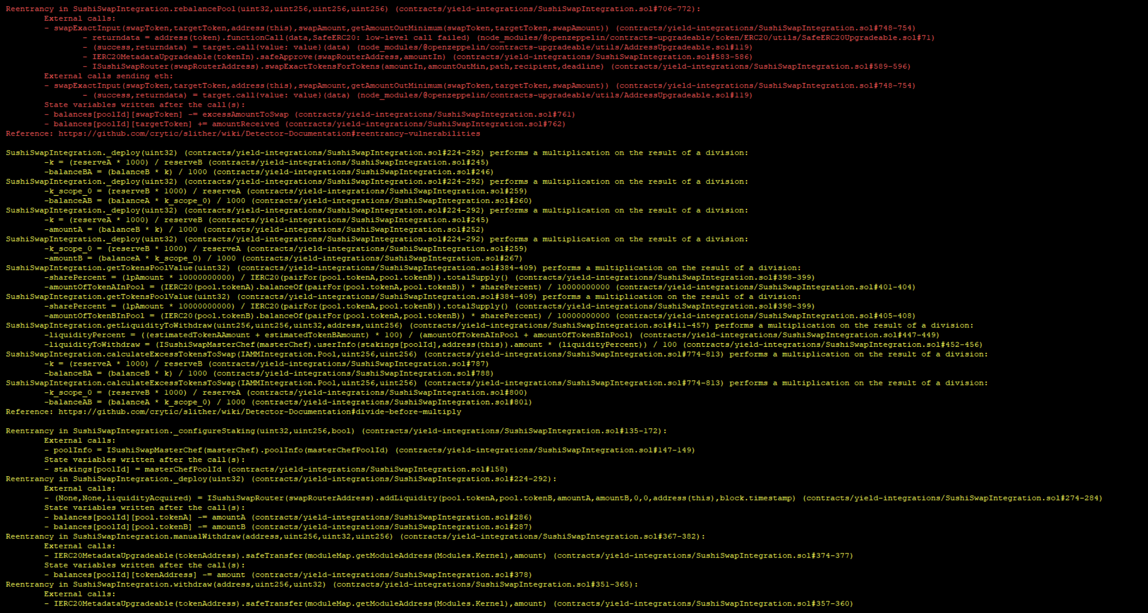


```
        External calls:
        - ISushiSwapMasterChefV2(masterChef).harvest(stakings[poolId],address(this)) (contracts/yield-integrations/SushiSwapIntegrationV2.sol#345-348)
        - amounts = swapExactInput(pool.tokenA,wethAddress,address(this),tokenADiff) (contracts/yield-integrations/SushiSwapIntegrationV2.sol#363-368)
                - returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/utils/SafeERC20Upgradeable.sol#71)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#119)
                - IERC20MetadataUpgradeable(tokenIn).safeApprove(swapRouterAddress,amountIn) (contracts/yield-integrations/SushiSwapIntegrationV2.sol#625-628)
                - ISushiSwapRouter(swapRouterAddress).swapExactTokensForTokens(amountIn,amountOutMin,path,recipient,deadline) (contracts/yield-integrations/SushiSwapIntegrationV2.sol#631-638)
        External calls sending eth:
        - amounts = swapExactInput(pool.tokenA,wethAddress,address(this),tokenADiff) (contracts/yield-integrations/SushiSwapIntegrationV2.sol#363-368)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#119)
        State variables written after the call(s):
        - balances[poolId][pool.tokenB] += tokenBDiff (contracts/yield-integrations/SushiSwapIntegrationV2.sol#374)
Reentrancy in SushiSwapIntegrationV2.harvestYieldByPool(uint32,uint256,uint256,uint256,bool) (contracts/yield-integrations/SushiSwapIntegrationV2.sol#320-404):
        External calls:
        - ISushiSwapMasterChefV2(masterChef).harvest(stakings[poolId],address(this)) (contracts/yield-integrations/SushiSwapIntegrationV2.sol#345-348)
        - amounts = swapExactInput(pool.tokenA,wethAddress,address(this),tokenADiff) (contracts/yield-integrations/SushiSwapIntegrationV2.sol#363-368)
                - returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/utils/SafeERC20Upgradeable.sol#71)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#119)
                - IERC20MetadataUpgradeable(tokenIn).safeApprove(swapRouterAddress,amountIn) (contracts/yield-integrations/SushiSwapIntegrationV2.sol#625-628)
                - ISushiSwapRouter(swapRouterAddress).swapExactTokensForTokens(amountIn,amountOutMin,path,recipient,deadline) (contracts/yield-integrations/SushiSwapIntegrationV2.sol#631-638)
        - amounts_scope_0 = swapExactInput(pool.tokenB,wethAddress,address(this),tokenBDiff) (contracts/yield-integrations/SushiSwapIntegrationV2.sol#377-382)
                - returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/utils/SafeERC20Upgradeable.sol#71)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#119)
                - IERC20MetadataUpgradeable(tokenIn).safeApprove(swapRouterAddress,amountIn) (contracts/yield-integrations/SushiSwapIntegrationV2.sol#625-628)
                - ISushiSwapRouter(swapRouterAddress).swapExactTokensForTokens(amountIn,amountOutMin,path,recipient,deadline) (contracts/yield-integrations/SushiSwapIntegrationV2.sol#631-638)
        - amounts_scope_1 = swapExactInput(sushi,wethAddress,address(this),sushiAmount) (contracts/yield-integrations/SushiSwapIntegrationV2.sol#392-397)
                - returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/utils/SafeERC20Upgradeable.sol#71)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#119)
                - IERC20MetadataUpgradeable(tokenIn).safeApprove(swapRouterAddress,amountIn) (contracts/yield-integrations/SushiSwapIntegrationV2.sol#625-628)
                - ISushiSwapRouter(swapRouterAddress).swapExactTokensForTokens(amountIn,amountOutMin,path,recipient,deadline) (contracts/yield-integrations/SushiSwapIntegrationV2.sol#631-638)
        External calls sending eth:
        - amounts = swapExactInput(pool.tokenA,wethAddress,address(this),tokenADiff) (contracts/yield-integrations/SushiSwapIntegrationV2.sol#363-368)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#119)
        - amounts_scope_0 = swapExactInput(pool.tokenB,wethAddress,address(this),tokenBDiff) (contracts/yield-integrations/SushiSwapIntegrationV2.sol#377-382)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#119)
        - amounts_scope_1 = swapExactInput(sushi,wethAddress,address(this),sushiAmount) (contracts/yield-integrations/SushiSwapIntegrationV2.sol#392-397)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#119)
        State variables written after the call(s):
        - yieldBalances[poolId] += yieldAmount (contracts/yield-integrations/SushiSwapIntegrationV2.sol#401)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2

Reentrancy in SushiSwapIntegrationV2._deploy(uint32) (contracts/yield-integrations/SushiSwapIntegrationV2.sol#221-289):
        External calls:
        - (None,None,liquidityAcquired) = ISushiSwapRouter(swapRouterAddress).addLiquidity(pool.tokenA,pool.tokenB,amountA,amountB,0,0,address(this),block.timestamp) (contracts/yield-integrations/SushiSwapIntegrationV2.sol#271-281)
        Event emitted after the call(s):
        - DepositToPool(poolId,liquidityAcquired) (contracts/yield-integrations/SushiSwapIntegrationV2.sol#287)
        - LPTokensAcquired(liquidityAcquired) (contracts/yield-integrations/SushiSwapIntegrationV2.sol#286)
Reentrancy in SushiSwapIntegrationV2._withdraw(address,uint256,uint32,uint256) (contracts/yield-integrations/SushiSwapIntegrationV2.sol#517-601):
        External calls:
        - ISushiSwapMasterChefV2(masterChef).withdraw(stakings[poolId],liquidityToWithdraw,address(this)) (contracts/yield-integrations/SushiSwapIntegrationV2.sol#542-546)
        - IERC20MetadataUpgradeable(pairFor(pool.tokenA,pool.tokenB)).safeApprove(swapRouterAddress,liquidityToWithdraw) (contracts/yield-integrations/SushiSwapIntegrationV2.sol#548-549)
        - (amountTokenA,amountTokenB) = ISushiSwapRouter(swapRouterAddress).removeLiquidity(pool.tokenA,pool.tokenB,liquidityToWithdraw,0,0,address(this),block.timestamp + 360) (contracts/yield-integrations/SushiSwapIntegrationV2.sol#551
-561)
        - amountsOfTokenReceived1 = swapExactInput(pool.tokenB,tokenAddress,address(this),amountTokenB) (contracts/yield-integrations/SushiSwapIntegrationV2.sol#567-572)
                - returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/utils/SafeERC20Upgradeable.sol#71)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#119)
                - IERC20MetadataUpgradeable(tokenIn).safeApprove(swapRouterAddress,amountIn) (contracts/yield-integrations/SushiSwapIntegrationV2.sol#625-628)
                - ISushiSwapRouter(swapRouterAddress).swapExactTokensForTokens(amountIn,amountOutMin,path,recipient,deadline) (contracts/yield-integrations/SushiSwapIntegrationV2.sol#631-638)
        - IERC20MetadataUpgradeable(pool.tokenA).safeTransfer(moduleMap.getModuleAddress(Modules.Kernel),amountTokenA + amountsOfTokenReceived1[1]) (contracts/yield-integrations/SushiSwapIntegrationV2.sol#574-577)
        External calls sending eth:
        - amountsOfTokenReceived1 = swapExactInput(pool.tokenB,tokenAddress,address(this),amountTokenB) (contracts/yield-integrations/SushiSwapIntegrationV2.sol#567-572)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#119)
        Event emitted after the call(s):

        - WithdrawnFromPool(poolId,amountsOfTokenReceived1[1] + amountTokenA) (contracts/yield-integrations/SushiSwapIntegrationV2.sol#579-582)
Reentrancy in SushiSwapIntegrationV2._withdraw(address,uint256,uint32,uint256) (contracts/yield-integrations/SushiSwapIntegrationV2.sol#517-601):
        External calls:
        - ISushiSwapMasterChefV2(masterChef).withdraw(stakings[poolId],liquidityToWithdraw,address(this)) (contracts/yield-integrations/SushiSwapIntegrationV2.sol#542-546)
        - IERC20MetadataUpgradeable(pairFor(pool.tokenA,pool.tokenB)).safeApprove(swapRouterAddress,liquidityToWithdraw) (contracts/yield-integrations/SushiSwapIntegrationV2.sol#548-549)
        - (amountTokenA,amountTokenB) = ISushiSwapRouter(swapRouterAddress).removeLiquidity(pool.tokenA,pool.tokenB,liquidityToWithdraw,0,0,address(this),block.timestamp + 360) (contracts/yield-integrations/SushiSwapIntegrationV2.sol#551
-561)
        - amountsOfTokenReceived0 = swapExactInput(pool.tokenA,tokenAddress,address(this),amountTokenA) (contracts/yield-integrations/SushiSwapIntegrationV2.sol#584-589)
                - returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/utils/SafeERC20Upgradeable.sol#71)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#119)
                - IERC20MetadataUpgradeable(tokenIn).safeApprove(swapRouterAddress,amountIn) (contracts/yield-integrations/SushiSwapIntegrationV2.sol#625-628)
                - ISushiSwapRouter(swapRouterAddress).swapExactTokensForTokens(amountIn,amountOutMin,path,recipient,deadline) (contracts/yield-integrations/SushiSwapIntegrationV2.sol#631-638)
        - IERC20MetadataUpgradeable(pool.tokenB).safeTransfer(moduleMap.getModuleAddress(Modules.Kernel),amountTokenB + amountsOfTokenReceived0[1]) (contracts/yield-integrations/SushiSwapIntegrationV2.sol#591-594)
        External calls sending eth:
        - amountsOfTokenReceived0 = swapExactInput(pool.tokenA,tokenAddress,address(this),amountTokenA) (contracts/yield-integrations/SushiSwapIntegrationV2.sol#584-589)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#119)
        Event emitted after the call(s):
        - WithdrawnFromPool(poolId,amountsOfTokenReceived0[1] + amountTokenB) (contracts/yield-integrations/SushiSwapIntegrationV2.sol#596-599)
Reentrancy in SushiSwapIntegrationV2.harvestYield() (contracts/yield-integrations/SushiSwapIntegrationV2.sol#305-317):
        External calls:
        - IERC20MetadataUpgradeable(wethAddress).safeTransfer(moduleMap.getModuleAddress(Modules.YieldManager),yieldAmount) (contracts/yield-integrations/SushiSwapIntegrationV2.sol#311-314)
        Event emitted after the call(s):
        - YieldReceived(yieldAmount) (contracts/yield-integrations/SushiSwapIntegrationV2.sol#315)
Reentrancy in SushiSwapIntegrationV2.harvestYieldByPool(uint32,uint256,uint256,uint256,bool) (contracts/yield-integrations/SushiSwapIntegrationV2.sol#320-404):
        External calls:
        - ISushiSwapMasterChefV2(masterChef).harvest(stakings[poolId],address(this)) (contracts/yield-integrations/SushiSwapIntegrationV2.sol#345-348)
        - amounts = swapExactInput(pool.tokenA,wethAddress,address(this),tokenADiff) (contracts/yield-integrations/SushiSwapIntegrationV2.sol#363-368)
                - returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/utils/SafeERC20Upgradeable.sol#71)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#119)
                - IERC20MetadataUpgradeable(tokenIn).safeApprove(swapRouterAddress,amountIn) (contracts/yield-integrations/SushiSwapIntegrationV2.sol#625-628)
                - ISushiSwapRouter(swapRouterAddress).swapExactTokensForTokens(amountIn,amountOutMin,path,recipient,deadline) (contracts/yield-integrations/SushiSwapIntegrationV2.sol#631-638)
        - amounts_scope_0 = swapExactInput(pool.tokenB,wethAddress,address(this),tokenBDiff) (contracts/yield-integrations/SushiSwapIntegrationV2.sol#377-382)
                - returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/utils/SafeERC20Upgradeable.sol#71)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#119)
                - IERC20MetadataUpgradeable(tokenIn).safeApprove(swapRouterAddress,amountIn) (contracts/yield-integrations/SushiSwapIntegrationV2.sol#625-628)
                - ISushiSwapRouter(swapRouterAddress).swapExactTokensForTokens(amountIn,amountOutMin,path,recipient,deadline) (contracts/yield-integrations/SushiSwapIntegrationV2.sol#631-638)
        - amounts_scope_1 = swapExactInput(sushi,wethAddress,address(this),sushiAmount) (contracts/yield-integrations/SushiSwapIntegrationV2.sol#392-397)
                - returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/utils/SafeERC20Upgradeable.sol#71)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#119)
                - IERC20MetadataUpgradeable(tokenIn).safeApprove(swapRouterAddress,amountIn) (contracts/yield-integrations/SushiSwapIntegrationV2.sol#625-628)
                - ISushiSwapRouter(swapRouterAddress).swapExactTokensForTokens(amountIn,amountOutMin,path,recipient,deadline) (contracts/yield-integrations/SushiSwapIntegrationV2.sol#631-638)
        External calls sending eth:
        - amounts = swapExactInput(pool.tokenA,wethAddress,address(this),tokenADiff) (contracts/yield-integrations/SushiSwapIntegrationV2.sol#363-368)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#119)
        - amounts_scope_0 = swapExactInput(pool.tokenB,wethAddress,address(this),tokenBDiff) (contracts/yield-integrations/SushiSwapIntegrationV2.sol#377-382)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#119)
        - amounts_scope_1 = swapExactInput(sushi,wethAddress,address(this),sushiAmount) (contracts/yield-integrations/SushiSwapIntegrationV2.sol#392-397)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#119)
        Event emitted after the call(s):
        - YieldReceived(yieldAmount) (contracts/yield-integrations/SushiSwapIntegrationV2.sol#403)
Reentrancy in SushiSwapIntegrationV2.rebalancePool(uint32,uint256,uint256,uint256) (contracts/yield-integrations/SushiSwapIntegrationV2.sol#763-823):
        External calls:
        - swapExactInput(swapToken,targetToken,address(this),swapAmount) (contracts/yield-integrations/SushiSwapIntegrationV2.sol#805)
                - returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/utils/SafeERC20Upgradeable.sol#71)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#119)
                - IERC20MetadataUpgradeable(tokenIn).safeApprove(swapRouterAddress,amountIn) (contracts/yield-integrations/SushiSwapIntegrationV2.sol#625-628)
                - ISushiSwapRouter(swapRouterAddress).swapExactTokensForTokens(amountIn,amountOutMin,path,recipient,deadline) (contracts/yield-integrations/SushiSwapIntegrationV2.sol#631-638)
        External calls sending eth:
        - swapExactInput(swapToken,targetToken,address(this),swapAmount) (contracts/yield-integrations/SushiSwapIntegrationV2.sol#805)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#119)
        Event emitted after the call(s):
        - PoolRebalanced(poolId,swapToken,excessAmountToSwap,targetToken,amountReceived) (contracts/yield-integrations/SushiSwapIntegrationV2.sol#815-821)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3


SushiSwapIntegrationV2.harvestYieldByPool(uint32,uint256,uint256,uint256,bool) (contracts/yield-integrations/SushiSwapIntegrationV2.sol#320-404) uses timestamp for comparisons
        Dangerous comparisons:
        - yieldAmount > 0 (contracts/yield-integrations/SushiSwapIntegrationV2.sol#400)
SushiSwapIntegrationV2.swapExactInput(address,address,address,uint256) (contracts/yield-integrations/SushiSwapIntegrationV2.sol#607-639) uses timestamp for comparisons
        Dangerous comparisons:
        - IERC20MetadataUpgradeable(tokenIn).allowance(address(this),swapRouterAddress) < amountIn (contracts/yield-integrations/SushiSwapIntegrationV2.sol#620-623)
SushiSwapIntegrationV2.getAmountOut(address,address,uint256) (contracts/yield-integrations/SushiSwapIntegrationV2.sol#660-678) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(amountIn > 0,amountIn must be greater than zero) (contracts/yield-integrations/SushiSwapIntegrationV2.sol#665)
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

SushiSwapIntegrationV2._configureStaking(uint32,uint256,bool) (contracts/yield-integrations/SushiSwapIntegrationV2.sol#135-169) compares to a boolean constant:
        -overwrite != true (contracts/yield-integrations/SushiSwapIntegrationV2.sol#143)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#boolean-equality

Different versions of Solidity is used:
        - Version used: ['0.8.4', '^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/interfaces/IAMMIntegration.sol#2)
        - 0.8.4 (contracts/interfaces/IERC20.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/ISushiSwapFactory.sol#2)
        - 0.8.4 (contracts/interfaces/ISushiSwapMasterChefV2.sol#2)
        - 0.8.4 (contracts/interfaces/ISushiSwapPair.sol#2)
        - 0.8.4 (contracts/interfaces/ISushiSwapRouter.sol#2)
        - 0.8.4 (contracts/interfaces/IWeth9.sol#2)
        - 0.8.4 (contracts/yield-integrations/SushiSwapIntegrationV2.sol#2)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used
```

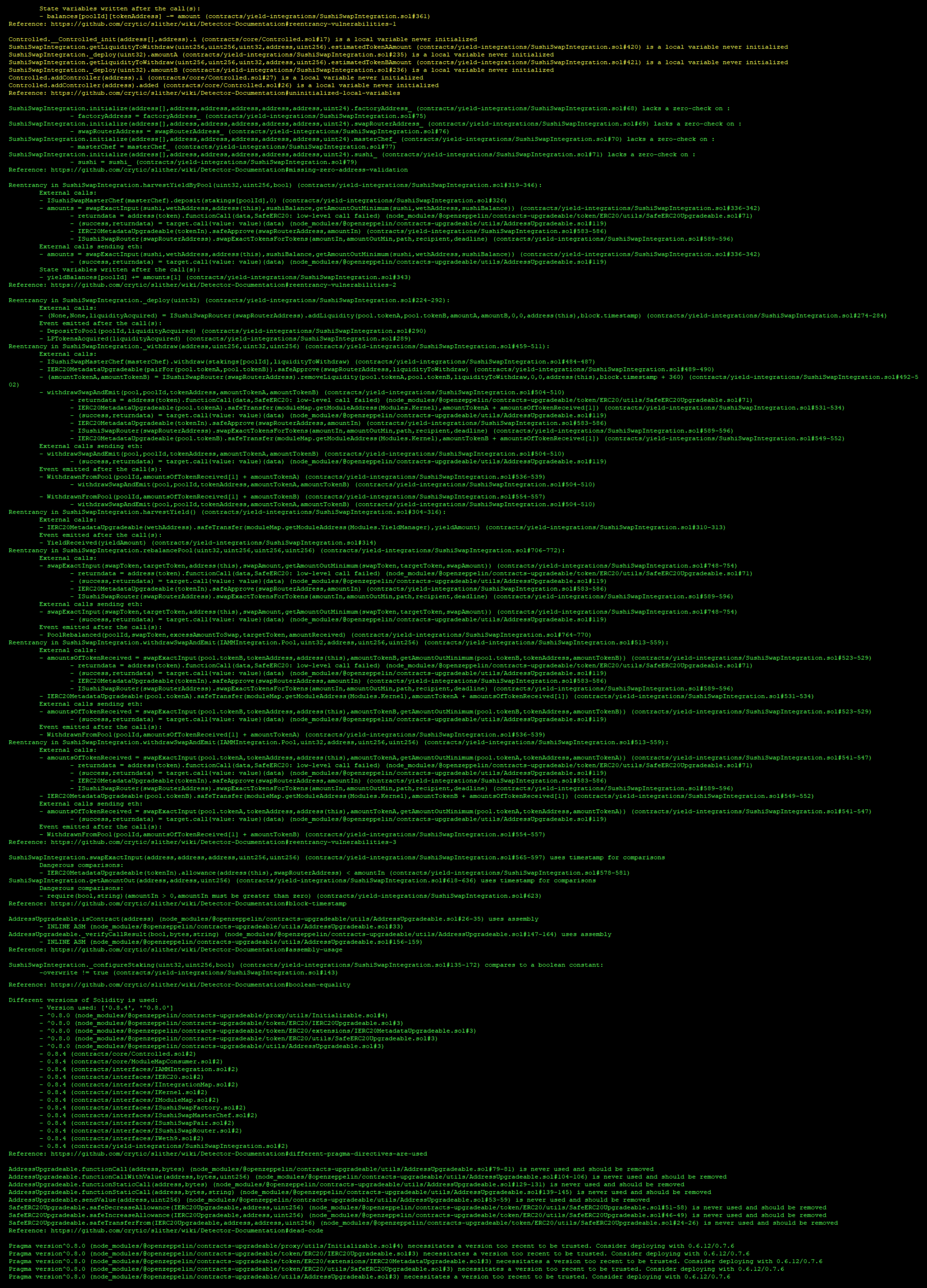


```
AddressUpgradeable.functionCall(address,bytes) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#79-81) is never used and should be removed
AddressUpgradeable.functionCallWithValue(address,bytes,uint256) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#104-106) 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.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.safeTransferFrom(IERC20Upgradeable,address,address,uint256) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/utils/SafeERC20Upgradeable.sol#24-26) 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/interfaces/IAMMIntegration.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/IERC20.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/ISushiSwapFactory.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/ISushiSwapMasterChefV2.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/ISushiSwapPair.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/ISushiSwapRouter.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/yield-integrations/SushiSwapIntegrationV2.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
Function ISushiSwapRouter.WETH() (contracts/interfaces/ISushiSwapRouter.sol#87) 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 IAMMIntegration.rebalancePool(uint32,uint256,uint256,uint256).maxSellTokenA (contracts/interfaces/IAMMIntegration.sol#69) is too similar to IAMMIntegration.rebalancePool(uint32,uint256,uint256,uint256).maxSellTokenB (contracts/i
nterfaces/IAMMIntegration.sol#70)
Variable ISushiSwapRouter.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (contracts/interfaces/ISushiSwapRouter.sol#24) is too similar to ISushiSwapRouter.addLiquidity(address,address,uint256
,uint256,uint256,uint256,address,uint256).amountBDesired (contracts/interfaces/ISushiSwapRouter.sol#25)
Variable Controlled._controllers (contracts/core/Controlled.sol#10) is too similar to SushiSwapIntegrationV2.initialize(address[],address,address,address,address,address,uint24).controllers_ (contracts/yield-integrations/SushiSwapIntegra
tionV2.sol#66)
Variable SushiSwapIntegrationV2.getLiquidityToWithdraw(uint256,uint256,uint32,address,uint256).amountOfTokenAInPool (contracts/yield-integrations/SushiSwapIntegrationV2.sol#453) is too similar to SushiSwapIntegrationV2.getLiquidityToWith
draw(uint256,uint256,uint32,address,uint256).amountOfTokenBInPool (contracts/yield-integrations/SushiSwapIntegrationV2.sol#454)
Variable SushiSwapIntegrationV2.getTokensPoolValue(uint32).amountOfTokenAInPool (contracts/yield-integrations/SushiSwapIntegrationV2.sol#428) is too similar to SushiSwapIntegrationV2._withdraw(address,uint256,uint32,uint256).amountOfToke
nBInPool (contracts/yield-integrations/SushiSwapIntegrationV2.sol#531)
Variable SushiSwapIntegrationV2.getLiquidityToWithdraw(uint256,uint256,uint32,address,uint256).amountOfTokenAInPool (contracts/yield-integrations/SushiSwapIntegrationV2.sol#453) is too similar to SushiSwapIntegrationV2._withdraw(address,
uint256,uint32,uint256).amountOfTokenBInPool (contracts/yield-integrations/SushiSwapIntegrationV2.sol#531)
Variable SushiSwapIntegrationV2._withdraw(address,uint256,uint32,uint256).amountOfTokenAInPool (contracts/yield-integrations/SushiSwapIntegrationV2.sol#530) is too similar to SushiSwapIntegrationV2._withdraw(address,uint256,uint32,uint25
6).amountOfTokenBInPool (contracts/yield-integrations/SushiSwapIntegrationV2.sol#531)
Variable SushiSwapIntegrationV2.getTokensPoolValue(uint32).amountOfTokenAInPool (contracts/yield-integrations/SushiSwapIntegrationV2.sol#428) is too similar to SushiSwapIntegrationV2.getTokensPoolValue(uint32).amountOfTokenBInPool (contr
acts/yield-integrations/SushiSwapIntegrationV2.sol#428)
Variable SushiSwapIntegrationV2.getLiquidityToWithdraw(uint256,uint256,uint32,address,uint256).amountOfTokenAInPool (contracts/yield-integrations/SushiSwapIntegrationV2.sol#453) is too similar to SushiSwapIntegrationV2.getTokensPoolValue
(uint32).amountOfTokenBInPool (contracts/yield-integrations/SushiSwapIntegrationV2.sol#428)
Variable SushiSwapIntegrationV2._withdraw(address,uint256,uint32,uint256).amountTokenA (contracts/yield-integrations/SushiSwapIntegrationV2.sol#551) is too similar to SushiSwapIntegrationV2._withdraw(address,uint256,uint32,uint256).amoun
tTokenB (contracts/yield-integrations/SushiSwapIntegrationV2.sol#551)
Variable SushiSwapIntegrationV2._withdraw(address,uint256,uint32,uint256).amountsOfTokenReceived0 (contracts/yield-integrations/SushiSwapIntegrationV2.sol#563) is too similar to SushiSwapIntegrationV2._withdraw(address,uint256,uint32,uin
t256).amountsOfTokenReceived1 (contracts/yield-integrations/SushiSwapIntegrationV2.sol#564)
Variable SushiSwapIntegrationV2.harvestYieldByPool(uint32,uint256,uint256,uint256,bool).amounts_scope_0 (contracts/yield-integrations/SushiSwapIntegrationV2.sol#377-382) is too similar to SushiSwapIntegrationV2.harvestYieldByPool(uint32,
uint256,uint256,uint256,bool).amounts_scope_1 (contracts/yield-integrations/SushiSwapIntegrationV2.sol#392-397)
Variable SushiSwapIntegrationV2.getLiquidityToWithdraw(uint256,uint256,uint32,address,uint256).estimatedTokenAAmount (contracts/yield-integrations/SushiSwapIntegrationV2.sol#461) is too similar to SushiSwapIntegrationV2.getLiquidityToWit
hdraw(uint256,uint256,uint32,address,uint256).estimatedTokenBAmount (contracts/yield-integrations/SushiSwapIntegrationV2.sol#462)
Variable SushiSwapIntegrationV2.rebalancePool(uint32,uint256,uint256,uint256).maxSellTokenA (contracts/yield-integrations/SushiSwapIntegrationV2.sol#766) is too similar to SushiSwapIntegrationV2.rebalancePool(uint32,uint256,uint256,uint2
56).maxSellTokenB (contracts/yield-integrations/SushiSwapIntegrationV2.sol#767)
Variable SushiSwapIntegrationV2.rebalancePool(uint32,uint256,uint256,uint256).maxSellTokenA (contracts/yield-integrations/SushiSwapIntegrationV2.sol#766) is too similar to IAMMIntegration.rebalancePool(uint32,uint256,uint256,uint256).max
SellTokenB (contracts/interfaces/IAMMIntegration.sol#70)
Variable IAMMIntegration.rebalancePool(uint32,uint256,uint256,uint256).maxSellTokenA (contracts/interfaces/IAMMIntegration.sol#69) is too similar to SushiSwapIntegrationV2.rebalancePool(uint32,uint256,uint256,uint256).maxSellTokenB (cont
racts/yield-integrations/SushiSwapIntegrationV2.sol#767)
Variable SushiSwapIntegrationV2.harvestYieldByPool(uint32,uint256,uint256,uint256,bool).tokenABalanceBefore (contracts/yield-integrations/SushiSwapIntegrationV2.sol#339-340) is too similar to SushiSwapIntegrationV2.harvestYieldByPool(uin
t32,uint256,uint256,uint256,bool).tokenBBalanceBefore (contracts/yield-integrations/SushiSwapIntegrationV2.sol#341-342)
Variable SushiSwapIntegrationV2.harvestYieldByPool(uint32,uint256,uint256,uint256,bool).tokenARatioX1000 (contracts/yield-integrations/SushiSwapIntegrationV2.sol#323) is too similar to SushiSwapIntegrationV2.harvestYieldByPool(uint32,uin
t256,uint256,uint256,bool).tokenBRatioX1000 (contracts/yield-integrations/SushiSwapIntegrationV2.sol#324)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar

SushiSwapIntegrationV2.getTokensPoolValue(uint32) (contracts/yield-integrations/SushiSwapIntegrationV2.sol#425-450) uses literals with too many digits:
        - sharePercent = (lpAmount * 10000000000) / IERC20(pairFor(pool.tokenA,pool.tokenB)).totalSupply() (contracts/yield-integrations/SushiSwapIntegrationV2.sol#439-440)
SushiSwapIntegrationV2.getTokensPoolValue(uint32) (contracts/yield-integrations/SushiSwapIntegrationV2.sol#425-450) uses literals with too many digits:
        - amountOfTokenAInPool = (IERC20(pool.tokenA).balanceOf(pairFor(pool.tokenA,pool.tokenB)) * sharePercent) / 10000000000 (contracts/yield-integrations/SushiSwapIntegrationV2.sol#442-445)
SushiSwapIntegrationV2.getTokensPoolValue(uint32) (contracts/yield-integrations/SushiSwapIntegrationV2.sol#425-450) uses literals with too many digits:
        - amountOfTokenBInPool = (IERC20(pool.tokenB).balanceOf(pairFor(pool.tokenA,pool.tokenB)) * sharePercent) / 10000000000 (contracts/yield-integrations/SushiSwapIntegrationV2.sol#446-449)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits

initialize(address[],address,address,address,address,address,uint24) should be declared external:
        - SushiSwapIntegrationV2.initialize(address[],address,address,address,address,address,uint24) (contracts/yield-integrations/SushiSwapIntegrationV2.sol#65-83)
withdraw(address,uint256,uint32) should be declared external:
        - SushiSwapIntegrationV2.withdraw(address,uint256,uint32) (contracts/yield-integrations/SushiSwapIntegrationV2.sol#409-423)
getTokensOrdered(address,address) should be declared external:
        - SushiSwapIntegrationV2.getTokensOrdered(address,address) (contracts/yield-integrations/SushiSwapIntegrationV2.sol#729-741)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

SushiSwapIntegration.sol



```
Reentrancy in SushiSwapIntegration.rebalancePool(uint32,uint256,uint256,uint256) (contracts/yield-integrations/SushiSwapIntegration.sol#706-772):
        External calls:
        - swapExactInput(swapToken,targetToken,address(this),swapAmount,getAmountOutMinimum(swapToken,targetToken,swapAmount)) (contracts/yield-integrations/SushiSwapIntegration.sol#748-754)
                - returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/utils/SafeERC20Upgradeable.sol#71)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#119)
                - IERC20MetadataUpgradeable(tokenIn).safeApprove(swapRouterAddress,amountIn) (contracts/yield-integrations/SushiSwapIntegration.sol#583-586)
                - ISushiSwapRouter(swapRouterAddress).swapExactTokensForTokens(amountIn,amountOutMin,path,recipient,deadline) (contracts/yield-integrations/SushiSwapIntegration.sol#589-596)
        External calls sending eth:
        - swapExactInput(swapToken,targetToken,address(this),swapAmount,getAmountOutMinimum(swapToken,targetToken,swapAmount)) (contracts/yield-integrations/SushiSwapIntegration.sol#748-754)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#119)
        State variables written after the call(s):
        - balances[poolId][swapToken] -= excessAmountToSwap (contracts/yield-integrations/SushiSwapIntegration.sol#761)
        - balances[poolId][targetToken] += amountReceived (contracts/yield-integrations/SushiSwapIntegration.sol#762)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities

SushiSwapIntegration._deploy(uint32) (contracts/yield-integrations/SushiSwapIntegration.sol#224-292) performs a multiplication on the result of a division:
        -k = (reserveA * 1000) / reserveB (contracts/yield-integrations/SushiSwapIntegration.sol#245)
        -balanceBA = (balanceB * k) / 1000 (contracts/yield-integrations/SushiSwapIntegration.sol#246)
SushiSwapIntegration._deploy(uint32) (contracts/yield-integrations/SushiSwapIntegration.sol#224-292) performs a multiplication on the result of a division:
        -k_scope_0 = (reserveB * 1000) / reserveA (contracts/yield-integrations/SushiSwapIntegration.sol#259)
        -balanceAB = (balanceA * k_scope_0) / 1000 (contracts/yield-integrations/SushiSwapIntegration.sol#260)
SushiSwapIntegration._deploy(uint32) (contracts/yield-integrations/SushiSwapIntegration.sol#224-292) performs a multiplication on the result of a division:
        -k = (reserveA * 1000) / reserveB (contracts/yield-integrations/SushiSwapIntegration.sol#245)
        -amountA = (balanceB * k) / 1000 (contracts/yield-integrations/SushiSwapIntegration.sol#252)
SushiSwapIntegration._deploy(uint32) (contracts/yield-integrations/SushiSwapIntegration.sol#224-292) performs a multiplication on the result of a division:
        -k_scope_0 = (reserveB * 1000) / reserveA (contracts/yield-integrations/SushiSwapIntegration.sol#259)
        -amountB = (balanceA * k_scope_0) / 1000 (contracts/yield-integrations/SushiSwapIntegration.sol#267)
SushiSwapIntegration.getTokensPoolValue(uint32) (contracts/yield-integrations/SushiSwapIntegration.sol#384-409) performs a multiplication on the result of a division:
        -sharePercent = (lpAmount * 10000000000) / IERC20(pairFor(pool.tokenA,pool.tokenB)).totalSupply() (contracts/yield-integrations/SushiSwapIntegration.sol#398-399)
        -amountOfTokenAInPool = (IERC20(pool.tokenA).balanceOf(pairFor(pool.tokenA,pool.tokenB)) * sharePercent) / 10000000000 (contracts/yield-integrations/SushiSwapIntegration.sol#401-404)
SushiSwapIntegration.getTokensPoolValue(uint32) (contracts/yield-integrations/SushiSwapIntegration.sol#384-409) performs a multiplication on the result of a division:
        -sharePercent = (lpAmount * 10000000000) / IERC20(pairFor(pool.tokenA,pool.tokenB)).totalSupply() (contracts/yield-integrations/SushiSwapIntegration.sol#398-399)
        -amountOfTokenBInPool = (IERC20(pool.tokenB).balanceOf(pairFor(pool.tokenA,pool.tokenB)) * sharePercent) / 10000000000 (contracts/yield-integrations/SushiSwapIntegration.sol#405-408)
SushiSwapIntegration.getLiquidityToWithdraw(uint256,uint256,uint32,address,uint256) (contracts/yield-integrations/SushiSwapIntegration.sol#411-457) performs a multiplication on the result of a division:
        -liquidityPercent = ((estimatedTokenAAmount + estimatedTokenBAmount) * 100) / (amountOfTokenAInPool + amountOfTokenBInPool) (contracts/yield-integrations/SushiSwapIntegration.sol#447-449)
        -liquidityToWithdraw = (ISushiSwapMasterChef(masterChef).userInfo(stakings[poolId],address(this)).amount * (liquidityPercent)) / 100 (contracts/yield-integrations/SushiSwapIntegration.sol#452-456)
SushiSwapIntegration.calculateExcessTokensToSwap(IAMMIntegration.Pool,uint256,uint256) (contracts/yield-integrations/SushiSwapIntegration.sol#774-813) performs a multiplication on the result of a division:
        -k = (reserveA * 1000) / reserveB (contracts/yield-integrations/SushiSwapIntegration.sol#787)
        -balanceBA = (balanceB * k) / 1000 (contracts/yield-integrations/SushiSwapIntegration.sol#788)
SushiSwapIntegration.calculateExcessTokensToSwap(IAMMIntegration.Pool,uint256,uint256) (contracts/yield-integrations/SushiSwapIntegration.sol#774-813) performs a multiplication on the result of a division:
        -k_scope_0 = (reserveB * 1000) / reserveA (contracts/yield-integrations/SushiSwapIntegration.sol#800)
        -balanceAB = (balanceA * k_scope_0) / 1000 (contracts/yield-integrations/SushiSwapIntegration.sol#801)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply

Reentrancy in SushiSwapIntegration._configureStaking(uint32,uint256,bool) (contracts/yield-integrations/SushiSwapIntegration.sol#135-172):
        External calls:
        - poolInfo = ISushiSwapMasterChef(masterChef).poolInfo(masterChefPoolId) (contracts/yield-integrations/SushiSwapIntegration.sol#147-149)
        State variables written after the call(s):
        - stakings[poolId] = masterChefPoolId (contracts/yield-integrations/SushiSwapIntegration.sol#158)
Reentrancy in SushiSwapIntegration._deploy(uint32) (contracts/yield-integrations/SushiSwapIntegration.sol#224-292):
        External calls:
        - (None,None,liquidityAcquired) = ISushiSwapRouter(swapRouterAddress).addLiquidity(pool.tokenA,pool.tokenB,amountA,amountB,0,0,address(this),block.timestamp) (contracts/yield-integrations/SushiSwapIntegration.sol#274-284)
        State variables written after the call(s):
        - balances[poolId][pool.tokenA] -= amountA (contracts/yield-integrations/SushiSwapIntegration.sol#286)
        - balances[poolId][pool.tokenB] -= amountB (contracts/yield-integrations/SushiSwapIntegration.sol#287)
Reentrancy in SushiSwapIntegration.manualWithdraw(address,uint256,uint32,uint256) (contracts/yield-integrations/SushiSwapIntegration.sol#367-382):
        External calls:
        - IERC20MetadataUpgradeable(tokenAddress).safeTransfer(moduleMap.getModuleAddress(Modules.Kernel),amount) (contracts/yield-integrations/SushiSwapIntegration.sol#374-377)
        State variables written after the call(s):
        - balances[poolId][tokenAddress] -= amount (contracts/yield-integrations/SushiSwapIntegration.sol#378)
Reentrancy in SushiSwapIntegration.withdraw(address,uint256,uint32) (contracts/yield-integrations/SushiSwapIntegration.sol#351-365):
        External calls:
        - IERC20MetadataUpgradeable(tokenAddress).safeTransfer(moduleMap.getModuleAddress(Modules.Kernel),amount) (contracts/yield-integrations/SushiSwapIntegration.sol#357-360)
```

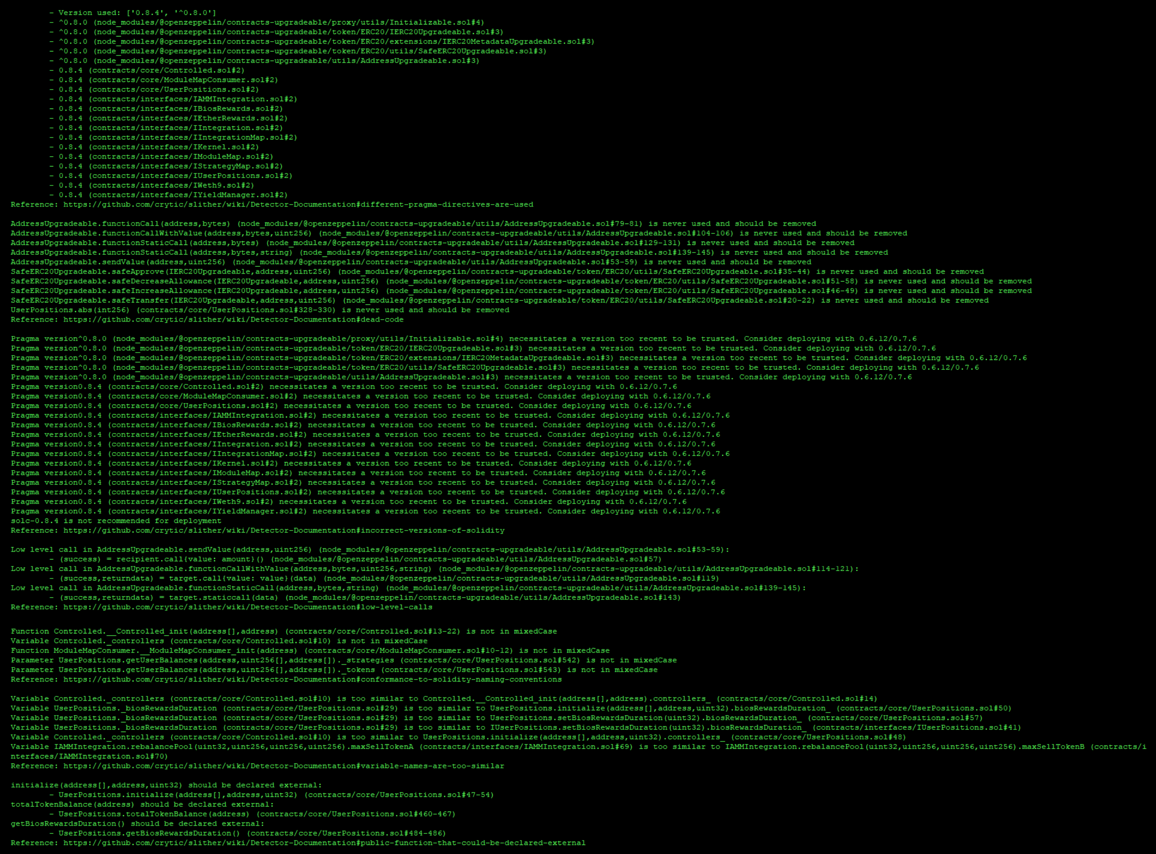


```
        State variables written after the call(s):
        - balances[poolId][tokenAddress] -= amount (contracts/yield-integrations/SushiSwapIntegration.sol#361)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1

Controlled.__Controlled_init(address[],address).i (contracts/core/Controlled.sol#17) is a local variable never initialized
SushiSwapIntegration.getLiquidityToWithdraw(uint256,uint256,uint32,address,uint256).estimatedTokenAAmount (contracts/yield-integrations/SushiSwapIntegration.sol#420) is a local variable never initialized
SushiSwapIntegration._deploy(uint32).amountA (contracts/yield-integrations/SushiSwapIntegration.sol#235) is a local variable never initialized
SushiSwapIntegration.getLiquidityToWithdraw(uint256,uint256,uint32,address,uint256).estimatedTokenBAmount (contracts/yield-integrations/SushiSwapIntegration.sol#421) is a local variable never initialized
SushiSwapIntegration._deploy(uint32).amountB (contracts/yield-integrations/SushiSwapIntegration.sol#236) 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

SushiSwapIntegration.initialize(address[],address,address,address,address,address,uint24).factoryAddress_ (contracts/yield-integrations/SushiSwapIntegration.sol#68) lacks a zero-check on :
                - factoryAddress = factoryAddress_ (contracts/yield-integrations/SushiSwapIntegration.sol#75)
SushiSwapIntegration.initialize(address[],address,address,address,address,address,uint24).swapRouterAddress_ (contracts/yield-integrations/SushiSwapIntegration.sol#69) lacks a zero-check on :
                - swapRouterAddress = swapRouterAddress_ (contracts/yield-integrations/SushiSwapIntegration.sol#76)
SushiSwapIntegration.initialize(address[],address,address,address,address,address,uint24).masterChef_ (contracts/yield-integrations/SushiSwapIntegration.sol#70) lacks a zero-check on :
                - masterChef = masterChef_ (contracts/yield-integrations/SushiSwapIntegration.sol#77)
SushiSwapIntegration.initialize(address[],address,address,address,address,address,uint24).sushi_ (contracts/yield-integrations/SushiSwapIntegration.sol#71) lacks a zero-check on :
                - sushi = sushi_ (contracts/yield-integrations/SushiSwapIntegration.sol#79)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

Reentrancy in SushiSwapIntegration.harvestYieldByPool(uint32,uint256,bool) (contracts/yield-integrations/SushiSwapIntegration.sol#319-346):
        External calls:
        - ISushiSwapMasterChef(masterChef).deposit(stakings[poolId],0) (contracts/yield-integrations/SushiSwapIntegration.sol#326)
        - amounts = swapExactInput(sushi,wethAddress,address(this),sushiBalance,getAmountOutMinimum(sushi,wethAddress,sushiBalance)) (contracts/yield-integrations/SushiSwapIntegration.sol#336-342)
                - returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/utils/SafeERC20Upgradeable.sol#71)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#119)
                - IERC20MetadataUpgradeable(tokenIn).safeApprove(swapRouterAddress,amountIn) (contracts/yield-integrations/SushiSwapIntegration.sol#583-586)
                - ISushiSwapRouter(swapRouterAddress).swapExactTokensForTokens(amountIn,amountOutMin,path,recipient,deadline) (contracts/yield-integrations/SushiSwapIntegration.sol#589-596)
        External calls sending eth:
        - amounts = swapExactInput(sushi,wethAddress,address(this),sushiBalance,getAmountOutMinimum(sushi,wethAddress,sushiBalance)) (contracts/yield-integrations/SushiSwapIntegration.sol#336-342)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#119)
        State variables written after the call(s):
        - yieldBalances[poolId] += amounts[1] (contracts/yield-integrations/SushiSwapIntegration.sol#343)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2

Reentrancy in SushiSwapIntegration._deploy(uint32) (contracts/yield-integrations/SushiSwapIntegration.sol#224-292):
        External calls:
        - (None,None,liquidityAcquired) = ISushiSwapRouter(swapRouterAddress).addLiquidity(pool.tokenA,pool.tokenB,amountA,amountB,0,0,address(this),block.timestamp) (contracts/yield-integrations/SushiSwapIntegration.sol#274-284)
        Event emitted after the call(s):
        - DepositToPool(poolId,liquidityAcquired) (contracts/yield-integrations/SushiSwapIntegration.sol#290)
        - LPTokensAcquired(liquidityAcquired) (contracts/yield-integrations/SushiSwapIntegration.sol#289)
Reentrancy in SushiSwapIntegration._withdraw(address,uint256,uint32,uint256) (contracts/yield-integrations/SushiSwapIntegration.sol#459-511):
        External calls:
        - ISushiSwapMasterChef(masterChef).withdraw(stakings[poolId],liquidityToWithdraw) (contracts/yield-integrations/SushiSwapIntegration.sol#484-487)
        - IERC20MetadataUpgradeable(pairFor(pool.tokenA,pool.tokenB)).safeApprove(swapRouterAddress,liquidityToWithdraw) (contracts/yield-integrations/SushiSwapIntegration.sol#489-490)
        - (amountTokenA,amountTokenB) = ISushiSwapRouter(swapRouterAddress).removeLiquidity(pool.tokenA,pool.tokenB,liquidityToWithdraw,0,0,address(this),block.timestamp + 360) (contracts/yield-integrations/SushiSwapIntegration.sol#492-5
02)
        - withdrawSwapAndEmit(pool,poolId,tokenAddress,amountTokenA,amountTokenB) (contracts/yield-integrations/SushiSwapIntegration.sol#504-510)
                - returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/utils/SafeERC20Upgradeable.sol#71)
                - IERC20MetadataUpgradeable(pool.tokenA).safeTransfer(moduleMap.getModuleAddress(Modules.Kernel),amountTokenA + amountsOfTokenReceived[1]) (contracts/yield-integrations/SushiSwapIntegration.sol#531-534)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#119)
                - IERC20MetadataUpgradeable(tokenIn).safeApprove(swapRouterAddress,amountIn) (contracts/yield-integrations/SushiSwapIntegration.sol#583-586)
                - ISushiSwapRouter(swapRouterAddress).swapExactTokensForTokens(amountIn,amountOutMin,path,recipient,deadline) (contracts/yield-integrations/SushiSwapIntegration.sol#589-596)
                - IERC20MetadataUpgradeable(pool.tokenB).safeTransfer(moduleMap.getModuleAddress(Modules.Kernel),amountTokenB + amountsOfTokenReceived[1]) (contracts/yield-integrations/SushiSwapIntegration.sol#549-552)
        External calls sending eth:
        - withdrawSwapAndEmit(pool,poolId,tokenAddress,amountTokenA,amountTokenB) (contracts/yield-integrations/SushiSwapIntegration.sol#504-510)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#119)
        Event emitted after the call(s):
        - WithdrawnFromPool(poolId,amountsOfTokenReceived[1] + amountTokenA) (contracts/yield-integrations/SushiSwapIntegration.sol#536-539)
                - withdrawSwapAndEmit(pool,poolId,tokenAddress,amountTokenA,amountTokenB) (contracts/yield-integrations/SushiSwapIntegration.sol#504-510)

        - WithdrawnFromPool(poolId,amountsOfTokenReceived[1] + amountTokenB) (contracts/yield-integrations/SushiSwapIntegration.sol#554-557)
                - withdrawSwapAndEmit(pool,poolId,tokenAddress,amountTokenA,amountTokenB) (contracts/yield-integrations/SushiSwapIntegration.sol#504-510)
Reentrancy in SushiSwapIntegration.harvestYield() (contracts/yield-integrations/SushiSwapIntegration.sol#304-316):
        External calls:
        - IERC20MetadataUpgradeable(wethAddress).safeTransfer(moduleMap.getModuleAddress(Modules.YieldManager),yieldAmount) (contracts/yield-integrations/SushiSwapIntegration.sol#310-313)
        Event emitted after the call(s):
        - YieldReceived(yieldAmount) (contracts/yield-integrations/SushiSwapIntegration.sol#314)
Reentrancy in SushiSwapIntegration.rebalancePool(uint32,uint256,uint256,uint256) (contracts/yield-integrations/SushiSwapIntegration.sol#706-772):
        External calls:
        - swapExactInput(swapToken,targetToken,address(this),swapAmount,getAmountOutMinimum(swapToken,targetToken,swapAmount)) (contracts/yield-integrations/SushiSwapIntegration.sol#748-754)
                - returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/utils/SafeERC20Upgradeable.sol#71)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#119)
                - IERC20MetadataUpgradeable(tokenIn).safeApprove(swapRouterAddress,amountIn) (contracts/yield-integrations/SushiSwapIntegration.sol#583-586)
                - ISushiSwapRouter(swapRouterAddress).swapExactTokensForTokens(amountIn,amountOutMin,path,recipient,deadline) (contracts/yield-integrations/SushiSwapIntegration.sol#589-596)
        External calls sending eth:
        - swapExactInput(swapToken,targetToken,address(this),swapAmount,getAmountOutMinimum(swapToken,targetToken,swapAmount)) (contracts/yield-integrations/SushiSwapIntegration.sol#748-754)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#119)
        Event emitted after the call(s):
        - PoolRebalanced(poolId,swapToken,excessAmountToSwap,targetToken,amountReceived) (contracts/yield-integrations/SushiSwapIntegration.sol#764-770)
Reentrancy in SushiSwapIntegration.withdrawSwapAndEmit(IAMMIntegration.Pool,uint32,address,uint256,uint256) (contracts/yield-integrations/SushiSwapIntegration.sol#513-559):
        External calls:
        - amountsOfTokenReceived = swapExactInput(pool.tokenB,tokenAddress,address(this),amountTokenB,getAmountOutMinimum(pool.tokenB,tokenAddress,amountTokenB)) (contracts/yield-integrations/SushiSwapIntegration.sol#523-529)
                - returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/utils/SafeERC20Upgradeable.sol#71)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#119)
                - IERC20MetadataUpgradeable(tokenIn).safeApprove(swapRouterAddress,amountIn) (contracts/yield-integrations/SushiSwapIntegration.sol#583-586)
                - ISushiSwapRouter(swapRouterAddress).swapExactTokensForTokens(amountIn,amountOutMin,path,recipient,deadline) (contracts/yield-integrations/SushiSwapIntegration.sol#589-596)
        - IERC20MetadataUpgradeable(pool.tokenA).safeTransfer(moduleMap.getModuleAddress(Modules.Kernel),amountTokenA + amountsOfTokenReceived[1]) (contracts/yield-integrations/SushiSwapIntegration.sol#531-534)
        External calls sending eth:
        - amountsOfTokenReceived = swapExactInput(pool.tokenB,tokenAddress,address(this),amountTokenB,getAmountOutMinimum(pool.tokenB,tokenAddress,amountTokenB)) (contracts/yield-integrations/SushiSwapIntegration.sol#523-529)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#119)
        Event emitted after the call(s):
        - WithdrawnFromPool(poolId,amountsOfTokenReceived[1] + amountTokenA) (contracts/yield-integrations/SushiSwapIntegration.sol#536-539)
Reentrancy in SushiSwapIntegration.withdrawSwapAndEmit(IAMMIntegration.Pool,uint32,address,uint256,uint256) (contracts/yield-integrations/SushiSwapIntegration.sol#513-559):
        External calls:
        - amountsOfTokenReceived = swapExactInput(pool.tokenA,tokenAddress,address(this),amountTokenA,getAmountOutMinimum(pool.tokenA,tokenAddress,amountTokenA)) (contracts/yield-integrations/SushiSwapIntegration.sol#541-547)
                - returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/utils/SafeERC20Upgradeable.sol#71)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#119)
                - IERC20MetadataUpgradeable(tokenIn).safeApprove(swapRouterAddress,amountIn) (contracts/yield-integrations/SushiSwapIntegration.sol#583-586)
                - ISushiSwapRouter(swapRouterAddress).swapExactTokensForTokens(amountIn,amountOutMin,path,recipient,deadline) (contracts/yield-integrations/SushiSwapIntegration.sol#589-596)
        - IERC20MetadataUpgradeable(pool.tokenB).safeTransfer(moduleMap.getModuleAddress(Modules.Kernel),amountTokenB + amountsOfTokenReceived[1]) (contracts/yield-integrations/SushiSwapIntegration.sol#549-552)
        External calls sending eth:
        - amountsOfTokenReceived = swapExactInput(pool.tokenA,tokenAddress,address(this),amountTokenA,getAmountOutMinimum(pool.tokenA,tokenAddress,amountTokenA)) (contracts/yield-integrations/SushiSwapIntegration.sol#541-547)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#119)
        Event emitted after the call(s):
        - WithdrawnFromPool(poolId,amountsOfTokenReceived[1] + amountTokenB) (contracts/yield-integrations/SushiSwapIntegration.sol#554-557)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3

SushiSwapIntegration.swapExactInput(address,address,address,uint256,uint256) (contracts/yield-integrations/SushiSwapIntegration.sol#565-597) uses timestamp for comparisons
        Dangerous comparisons:
        - IERC20MetadataUpgradeable(tokenIn).allowance(address(this),swapRouterAddress) < amountIn (contracts/yield-integrations/SushiSwapIntegration.sol#578-581)
SushiSwapIntegration.getAmountOut(address,address,uint256) (contracts/yield-integrations/SushiSwapIntegration.sol#618-636) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(amountIn > 0,amountIn must be greater than zero) (contracts/yield-integrations/SushiSwapIntegration.sol#623)
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

SushiSwapIntegration._configureStaking(uint32,uint256,bool) (contracts/yield-integrations/SushiSwapIntegration.sol#135-172) compares to a boolean constant:
        -overwrite != true (contracts/yield-integrations/SushiSwapIntegration.sol#143)

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#boolean-equality

Different versions of Solidity is used:
        - Version used: ['0.8.4', '^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/interfaces/IAMMIntegration.sol#2)
        - 0.8.4 (contracts/interfaces/IERC20.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/ISushiSwapFactory.sol#2)
        - 0.8.4 (contracts/interfaces/ISushiSwapMasterChef.sol#2)
        - 0.8.4 (contracts/interfaces/ISushiSwapPair.sol#2)
        - 0.8.4 (contracts/interfaces/ISushiSwapRouter.sol#2)
        - 0.8.4 (contracts/interfaces/IWeth9.sol#2)
        - 0.8.4 (contracts/yield-integrations/SushiSwapIntegration.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.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.safeTransferFrom(IERC20Upgradeable,address,address,uint256) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/utils/SafeERC20Upgradeable.sol#24-26) 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/interfaces/IAMMIntegration.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/IERC20.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/ISushiSwapFactory.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/ISushiSwapMasterChef.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/ISushiSwapPair.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/ISushiSwapRouter.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/yield-integrations/SushiSwapIntegration.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
Function ISushiSwapRouter.WETH() (contracts/interfaces/ISushiSwapRouter.sol#87) 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 IAMMIntegration.rebalancePool(uint32,uint256,uint256,uint256).maxSellTokenA (contracts/interfaces/IAMMIntegration.sol#69) is too similar to IAMMIntegration.rebalancePool(uint32,uint256,uint256,uint256).maxSellTokenB (contracts/i
nterfaces/IAMMIntegration.sol#70)
Variable ISushiSwapRouter.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (contracts/interfaces/ISushiSwapRouter.sol#24) is too similar to ISushiSwapRouter.addLiquidity(address,address,uint256
,uint256,uint256,uint256,address,uint256).amountBDesired (contracts/interfaces/ISushiSwapRouter.sol#25)
Variable SushiSwapIntegration._withdraw(address,uint256,uint32,uint256).amountOfTokenAInPool (contracts/yield-integrations/SushiSwapIntegration.sol#472) is too similar to SushiSwapIntegration.getTokensPoolValue(uint32).amountOfTokenBInPo
ol (contracts/yield-integrations/SushiSwapIntegration.sol#387)
Variable SushiSwapIntegration._withdraw(address,uint256,uint32,uint256).amountOfTokenAInPool (contracts/yield-integrations/SushiSwapIntegration.sol#472) is too similar to SushiSwapIntegration.getLiquidityToWithdraw(uint256,uint256,uint32
,address,uint256).amountOfTokenBInPool (contracts/yield-integrations/SushiSwapIntegration.sol#413)
Variable SushiSwapIntegration._withdraw(address,uint256,uint32,uint256).amountOfTokenAInPool (contracts/yield-integrations/SushiSwapIntegration.sol#472) is too similar to SushiSwapIntegration._withdraw(address,uint256,uint32,uint256).amo
untOfTokenBInPool (contracts/yield-integrations/SushiSwapIntegration.sol#473)
Variable SushiSwapIntegration.getTokensPoolValue(uint32).amountOfTokenAInPool (contracts/yield-integrations/SushiSwapIntegration.sol#387) is too similar to SushiSwapIntegration.getTokensPoolValue(uint32).amountOfTokenBInPool (contracts/y
ield-integrations/SushiSwapIntegration.sol#387)
Variable SushiSwapIntegration.getTokensPoolValue(uint32).amountOfTokenAInPool (contracts/yield-integrations/SushiSwapIntegration.sol#387) is too similar to SushiSwapIntegration.getLiquidityToWithdraw(uint256,uint256,uint32,address,uint25
6).amountOfTokenBInPool (contracts/yield-integrations/SushiSwapIntegration.sol#413)
Variable SushiSwapIntegration.getLiquidityToWithdraw(uint256,uint256,uint32,address,uint256).amountOfTokenAInPool (contracts/yield-integrations/SushiSwapIntegration.sol#412) is too similar to SushiSwapIntegration.getLiquidityToWithdraw(u
int256,uint256,uint32,address,uint256).amountOfTokenBInPool (contracts/yield-integrations/SushiSwapIntegration.sol#413)
Variable SushiSwapIntegration.getTokensPoolValue(uint32).amountOfTokenAInPool (contracts/yield-integrations/SushiSwapIntegration.sol#387) is too similar to SushiSwapIntegration._withdraw(address,uint256,uint32,uint256).amountOfTokenBInPo
ol (contracts/yield-integrations/SushiSwapIntegration.sol#473)
Variable SushiSwapIntegration.getLiquidityToWithdraw(uint256,uint256,uint32,address,uint256).amountOfTokenAInPool (contracts/yield-integrations/SushiSwapIntegration.sol#412) is too similar to SushiSwapIntegration.getTokensPoolValue(uint3
2).amountOfTokenBInPool (contracts/yield-integrations/SushiSwapIntegration.sol#387)
Variable SushiSwapIntegration.getLiquidityToWithdraw(uint256,uint256,uint32,address,uint256).amountOfTokenAInPool (contracts/yield-integrations/SushiSwapIntegration.sol#412) is too similar to SushiSwapIntegration._withdraw(address,uint25
6,uint32,uint256).amountOfTokenBInPool (contracts/yield-integrations/SushiSwapIntegration.sol#473)
Variable SushiSwapIntegration._withdraw(address,uint256,uint32,uint256).amountTokenA (contracts/yield-integrations/SushiSwapIntegration.sol#492) is too similar to SushiSwapIntegration._withdraw(address,uint256,uint32,uint256).amountToken
B (contracts/yield-integrations/SushiSwapIntegration.sol#492)
Variable SushiSwapIntegration.withdrawSwapAndEmit(IAMMIntegration.Pool,uint32,address,uint256,uint256).amountTokenA (contracts/yield-integrations/SushiSwapIntegration.sol#517) is too similar to SushiSwapIntegration.withdrawSwapAndEmit(IA
MMIntegration.Pool,uint32,address,uint256,uint256).amountTokenB (contracts/yield-integrations/SushiSwapIntegration.sol#518)
Variable SushiSwapIntegration.withdrawSwapAndEmit(IAMMIntegration.Pool,uint32,address,uint256,uint256).amountTokenA (contracts/yield-integrations/SushiSwapIntegration.sol#517) is too similar to SushiSwapIntegration._withdraw(address,uint
256,uint32,uint256).amountTokenB (contracts/yield-integrations/SushiSwapIntegration.sol#492)
Variable SushiSwapIntegration._withdraw(address,uint256,uint32,uint256).amountTokenA (contracts/yield-integrations/SushiSwapIntegration.sol#492) is too similar to SushiSwapIntegration.withdrawSwapAndEmit(IAMMIntegration.Pool,uint32,addre
ss,uint256,uint256).amountTokenB (contracts/yield-integrations/SushiSwapIntegration.sol#518)
Variable Controlled._controllers (contracts/core/Controlled.sol#10) is too similar to SushiSwapIntegration.initialize(address[],address,address,address,address,address,uint24).controllers_ (contracts/yield-integrations/SushiSwapIntegrati
on.sol#66)
```

UserPositions.sol

```
Reentrancy in UserPositions._withdraw(address,address[],uint256[],bool) (contracts/core/UserPositions.sol#252-326):
        External calls:
        - IStrategyMap(moduleMap.getModuleAddress(Modules.StrategyMap)).closePositionsForWithdrawal(tokens[tokenId],amounts[tokenId]) (contracts/core/UserPositions.sol#285-289)
        - IERC20MetadataUpgradeable(tokens[tokenId]).safeTransferFrom(moduleMap.getModuleAddress(Modules.Kernel),recipient,currentReserves) (contracts/core/UserPositions.sol#300-304)
        - IERC20MetadataUpgradeable(tokens[tokenId]).safeTransferFrom(moduleMap.getModuleAddress(Modules.Kernel),recipient,amounts[tokenId]) (contracts/core/UserPositions.sol#307-311)
        - IBiosRewards(moduleMap.getModuleAddress(Modules.BiosRewards)).decreaseRewards(tokens[tokenId],recipient,amounts[tokenId]) (contracts/core/UserPositions.sol#316-317)
        - IEtherRewards(moduleMap.getModuleAddress(Modules.EtherRewards)).updateUserRewards(tokens[tokenId],recipient) (contracts/core/UserPositions.sol#319-320)
        State variables written after the call(s):
        - _balances[tokens[tokenId]][recipient] -= amounts[tokenId] (contracts/core/UserPositions.sol#324)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1

Controlled.addController(address).added (contracts/core/Controlled.sol#26) is a local variable never initialized
UserPositions.withdrawAllAndClaim(address,address[],bool).tokenId (contracts/core/UserPositions.sol#223) is a local variable never initialized
UserPositions._withdraw(address,address[],uint256[],bool).tokenId (contracts/core/UserPositions.sol#268) is a local variable never initialized
Controlled.__Controlled_init(address[],address).i (contracts/core/Controlled.sol#17) is a local variable never initialized
UserPositions._increaseBiosRewards().tokenId (contracts/core/UserPositions.sol#362) is a local variable never initialized
Controlled.addController(address).i (contracts/core/Controlled.sol#27) is a local variable never initialized
UserPositions.deposit(address,address[],uint256[],uint256,bool).tokenId (contracts/core/UserPositions.sol#118) is a local variable never initialized
UserPositions._claimBiosRewards(address).tokenId (contracts/core/UserPositions.sol#440) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables

UserPositions.deposit(address,address[],uint256[],uint256,bool) (contracts/core/UserPositions.sol#105-182) has external calls inside a loop: require(bool,string)(integrationMap.getTokenAcceptingDeposits(tokens[tokenId]),UserPositions::de
posit: This token is not accepting deposits) (contracts/core/UserPositions.sol#120-123)
UserPositions.deposit(address,address[],uint256[],uint256,bool) (contracts/core/UserPositions.sol#105-182) has external calls inside a loop: beforeBalance = erc20.balanceOf(moduleMap.getModuleAddress(Modules.Kernel)) (contracts/core/User
Positions.sol#137-139)
UserPositions.deposit(address,address[],uint256[],uint256,bool) (contracts/core/UserPositions.sol#105-182) has external calls inside a loop: erc20.safeTransferFrom(depositor,moduleMap.getModuleAddress(Modules.Kernel),amounts[tokenId]) (c
ontracts/core/UserPositions.sol#142-146)
UserPositions.deposit(address,address[],uint256[],uint256,bool) (contracts/core/UserPositions.sol#105-182) has external calls inside a loop: afterBalance = erc20.balanceOf(moduleMap.getModuleAddress(Modules.Kernel)) (contracts/core/UserP
ositions.sol#149-151)
UserPositions.deposit(address,address[],uint256[],uint256,bool) (contracts/core/UserPositions.sol#105-182) has external calls inside a loop: IBiosRewards(moduleMap.getModuleAddress(Modules.BiosRewards)).increaseRewards(tokens[tokenId],de
positor,actualAmount) (contracts/core/UserPositions.sol#155-156)
UserPositions.deposit(address,address[],uint256[],uint256,bool) (contracts/core/UserPositions.sol#105-182) has external calls inside a loop: IEtherRewards(moduleMap.getModuleAddress(Modules.EtherRewards)).updateUserRewards(tokens[tokenId
],depositor) (contracts/core/UserPositions.sol#157-158)
UserPositions._withdraw(address,address[],uint256[],bool) (contracts/core/UserPositions.sol#252-326) has external calls inside a loop: require(bool,string)(integrationMap.getTokenAcceptingWithdrawals(tokens[tokenId]),UserPositions::_with
draw: This token is not accepting withdrawals) (contracts/core/UserPositions.sol#270-273)
UserPositions._withdraw(address,address[],uint256[],bool) (contracts/core/UserPositions.sol#252-326) has external calls inside a loop: reserveBalance = IERC20MetadataUpgradeable(tokens[tokenId]).balanceOf(moduleMap.getModuleAddress(Modul
es.Kernel)) (contracts/core/UserPositions.sol#281-282)
UserPositions._withdraw(address,address[],uint256[],bool) (contracts/core/UserPositions.sol#252-326) has external calls inside a loop: IStrategyMap(moduleMap.getModuleAddress(Modules.StrategyMap)).closePositionsForWithdrawal(tokens[token
Id],amounts[tokenId]) (contracts/core/UserPositions.sol#285-289)
UserPositions._withdraw(address,address[],uint256[],bool) (contracts/core/UserPositions.sol#252-326) has external calls inside a loop: IBiosRewards(moduleMap.getModuleAddress(Modules.BiosRewards)).decreaseRewards(tokens[tokenId],recipien
t,amounts[tokenId]) (contracts/core/UserPositions.sol#316-317)
UserPositions._withdraw(address,address[],uint256[],bool) (contracts/core/UserPositions.sol#252-326) has external calls inside a loop: IEtherRewards(moduleMap.getModuleAddress(Modules.EtherRewards)).updateUserRewards(tokens[tokenId],reci
pient) (contracts/core/UserPositions.sol#319-320)
UserPositions._withdraw(address,address[],uint256[],bool) (contracts/core/UserPositions.sol#252-326) has external calls inside a loop: currentReserves = IERC20MetadataUpgradeable(tokens[tokenId]).balanceOf(moduleMap.getModuleAddress(Modu
les.Kernel)) (contracts/core/UserPositions.sol#295-297)
UserPositions._withdraw(address,address[],uint256[],bool) (contracts/core/UserPositions.sol#252-326) has external calls inside a loop: IERC20MetadataUpgradeable(tokens[tokenId]).safeTransferFrom(moduleMap.getModuleAddress(Modules.Kernel)
,recipient,currentReserves) (contracts/core/UserPositions.sol#300-304)
UserPositions._withdraw(address,address[],uint256[],bool) (contracts/core/UserPositions.sol#252-326) has external calls inside a loop: IERC20MetadataUpgradeable(tokens[tokenId]).safeTransferFrom(moduleMap.getModuleAddress(Modules.Kernel)
,recipient,amounts[tokenId]) (contracts/core/UserPositions.sol#307-311)
UserPositions._increaseBiosRewards() (contracts/core/UserPositions.sol#338-370) has external calls inside a loop: token = integrationMap.getTokenAddress(tokenId) (contracts/core/UserPositions.sol#363)
UserPositions._increaseBiosRewards() (contracts/core/UserPositions.sol#338-370) has external calls inside a loop: tokenBiosRewardWeight = integrationMap.getTokenBiosRewardWeight(token) (contracts/core/UserPositions.sol#364-365)
UserPositions._claimBiosRewards(address) (contracts/core/UserPositions.sol#427-456) has external calls inside a loop: token = integrationMap.getTokenAddress(tokenId) (contracts/core/UserPositions.sol#441)
UserPositions._claimBiosRewards(address) (contracts/core/UserPositions.sol#427-456) has external calls inside a loop: biosRewards.earned(token,recipient) > 0 (contracts/core/UserPositions.sol#443)
UserPositions._claimBiosRewards(address) (contracts/core/UserPositions.sol#427-456) has external calls inside a loop: biosClaimed += IBiosRewards(moduleMap.getModuleAddress(Modules.BiosRewards)).claimReward(token,recipient) (contracts/co
re/UserPositions.sol#444-446)
UserPositions.getUserBalances(address,uint256[],address[]) (contracts/core/UserPositions.sol#540-581) has external calls inside a loop: strategyTokens = IStrategyMap(strategyMapAddress).getStrategy(_strategies[i_scope_0]).tokens (contrac
ts/core/UserPositions.sol#566-568)
UserPositions.migrateUser(uint256,IUserPositions.MigrateStrategy[]) (contracts/core/UserPositions.sol#612-624) has external calls inside a loop: strategyMap.increaseTokenBalance(strategyId,users[i].tokens) (contracts/core/UserPositions.s
ol#622)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-inside-a-loop

Reentrancy in UserPositions._withdraw(address,address[],uint256[],bool) (contracts/core/UserPositions.sol#252-326):
        External calls:

        - IStrategyMap(moduleMap.getModuleAddress(Modules.StrategyMap)).closePositionsForWithdrawal(tokens[tokenId],amounts[tokenId]) (contracts/core/UserPositions.sol#285-289)
        - IERC20MetadataUpgradeable(tokens[tokenId]).safeTransferFrom(moduleMap.getModuleAddress(Modules.Kernel),recipient,currentReserves) (contracts/core/UserPositions.sol#300-304)
        - IERC20MetadataUpgradeable(tokens[tokenId]).safeTransferFrom(moduleMap.getModuleAddress(Modules.Kernel),recipient,amounts[tokenId]) (contracts/core/UserPositions.sol#307-311)
        - IBiosRewards(moduleMap.getModuleAddress(Modules.BiosRewards)).decreaseRewards(tokens[tokenId],recipient,amounts[tokenId]) (contracts/core/UserPositions.sol#316-317)
        - IEtherRewards(moduleMap.getModuleAddress(Modules.EtherRewards)).updateUserRewards(tokens[tokenId],recipient) (contracts/core/UserPositions.sol#319-320)
        State variables written after the call(s):
        - _totalSupply[tokens[tokenId]] -= amounts[tokenId] (contracts/core/UserPositions.sol#323)
Reentrancy in UserPositions.deposit(address,address[],uint256[],uint256,bool) (contracts/core/UserPositions.sol#105-182):
        External calls:
        - erc20.safeTransferFrom(depositor,moduleMap.getModuleAddress(Modules.Kernel),amounts[tokenId]) (contracts/core/UserPositions.sol#142-146)
        - IBiosRewards(moduleMap.getModuleAddress(Modules.BiosRewards)).increaseRewards(tokens[tokenId],depositor,actualAmount) (contracts/core/UserPositions.sol#155-156)
        - IEtherRewards(moduleMap.getModuleAddress(Modules.EtherRewards)).updateUserRewards(tokens[tokenId],depositor) (contracts/core/UserPositions.sol#157-158)
        State variables written after the call(s):
        - _balances[tokens[tokenId]][depositor] += actualAmount (contracts/core/UserPositions.sol#164)
        - _totalSupply[tokens[tokenId]] += actualAmount (contracts/core/UserPositions.sol#163)
Reentrancy in UserPositions.deposit(address,address[],uint256[],uint256,bool) (contracts/core/UserPositions.sol#105-182):
        External calls:
        - IBiosRewards(moduleMap.getModuleAddress(Modules.BiosRewards)).increaseRewards(wethAddress,depositor,ethAmount) (contracts/core/UserPositions.sol#171-172)
        - IEtherRewards(moduleMap.getModuleAddress(Modules.EtherRewards)).updateUserRewards(wethAddress,depositor) (contracts/core/UserPositions.sol#173-174)
        State variables written after the call(s):
        - _balances[wethAddress][depositor] += ethAmount (contracts/core/UserPositions.sol#178)
        - _totalSupply[wethAddress] += ethAmount (contracts/core/UserPositions.sol#177)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2

Reentrancy in UserPositions.deposit(address,address[],uint256[],uint256,bool) (contracts/core/UserPositions.sol#105-182):
        External calls:
        - IBiosRewards(moduleMap.getModuleAddress(Modules.BiosRewards)).increaseRewards(wethAddress,depositor,ethAmount) (contracts/core/UserPositions.sol#171-172)
        - IEtherRewards(moduleMap.getModuleAddress(Modules.EtherRewards)).updateUserRewards(wethAddress,depositor) (contracts/core/UserPositions.sol#173-174)
        Event emitted after the call(s):
        - Deposit(depositor,tokens,actualAmounts,ethAmount) (contracts/core/UserPositions.sol#181)
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

Different versions of Solidity is used:
```

AUTOMATED TESTING



```
        - Version used: ['0.8.4', '^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/UserPositions.sol#2)
        - 0.8.4 (contracts/interfaces/IAMMIntegration.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/IUserPositions.sol#2)
        - 0.8.4 (contracts/interfaces/IWeth9.sol#2)
        - 0.8.4 (contracts/interfaces/IYieldManager.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
SafeERC20Upgradeable.safeTransfer(IERC20Upgradeable,address,uint256) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/utils/SafeERC20Upgradeable.sol#20-22) is never used and should be removed
UserPositions.abs(int256) (contracts/core/UserPositions.sol#328-330) 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/UserPositions.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/IAMMIntegration.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/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
Parameter UserPositions.getUserBalances(address,uint256[],address[])._strategies (contracts/core/UserPositions.sol#542) is not in mixedCase
Parameter UserPositions.getUserBalances(address,uint256[],address[])._tokens (contracts/core/UserPositions.sol#543) 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 UserPositions._biosRewardsDuration (contracts/core/UserPositions.sol#29) is too similar to UserPositions.initialize(address[],address,uint32).biosRewardsDuration_ (contracts/core/UserPositions.sol#50)
Variable UserPositions._biosRewardsDuration (contracts/core/UserPositions.sol#29) is too similar to UserPositions.setBiosRewardsDuration(uint32).biosRewardsDuration_ (contracts/core/UserPositions.sol#57)
Variable UserPositions._biosRewardsDuration (contracts/core/UserPositions.sol#29) is too similar to IUserPositions.setBiosRewardsDuration(uint32).biosRewardsDuration_ (contracts/interfaces/IUserPositions.sol#41)
Variable Controlled._controllers (contracts/core/Controlled.sol#10) is too similar to UserPositions.initialize(address[],address,uint32).controllers_ (contracts/core/UserPositions.sol#48)
Variable IAMMIntegration.rebalancePool(uint32,uint256,uint256,uint256).maxSellTokenA (contracts/interfaces/IAMMIntegration.sol#69) is too similar to IAMMIntegration.rebalancePool(uint32,uint256,uint256,uint256).maxSellTokenB (contracts/i
nterfaces/IAMMIntegration.sol#70)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar

initialize(address[],address,uint32) should be declared external:
        - UserPositions.initialize(address[],address,uint32) (contracts/core/UserPositions.sol#47-54)
totalTokenBalance(address) should be declared external:
        - UserPositions.totalTokenBalance(address) (contracts/core/UserPositions.sol#460-467)
getBiosRewardsDuration() should be declared external:
        - UserPositions.getBiosRewardsDuration() (contracts/core/UserPositions.sol#484-486)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

- No major issues found by Slither.

4.2 AUTOMATED SECURITY SCAN

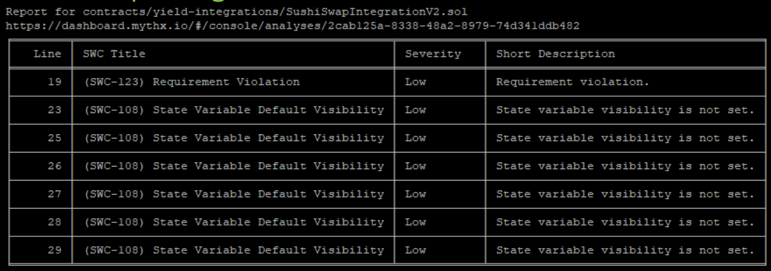
Description:

Halborn used automated security scanners to assist with detection of well-known security issues, and to identify low-hanging fruits on the targets for this engagement. Among the tools used was MythX, a security analysis service for Ethereum smart contracts. MythX performed a scan on all the contracts and sent the compiled results to the analyzers to locate any vulnerabilities.

MythX results:

MythX only found some issues in the following smart contracts:

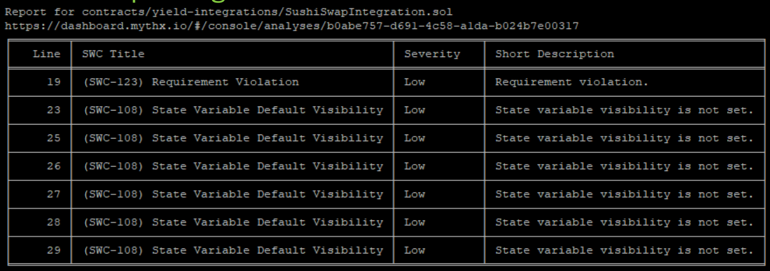
SushiSwapIntegrationV2.sol



Report for contracts/yield-integrations/SushiSwapIntegrationV2.sol
https://dashboard.mythx.io/#/console/analyses/2cab125a-8338-48a2-8979-74d341ddb482

Line	SWC Title	Severity	Short Description
19	(SWC-123) Requirement Violation	Low	Requirement violation.
23	(SWC-108) State Variable Default Visibility	Low	State variable visibility is not set.
25	(SWC-108) State Variable Default Visibility	Low	State variable visibility is not set.
26	(SWC-108) State Variable Default Visibility	Low	State variable visibility is not set.
27	(SWC-108) State Variable Default Visibility	Low	State variable visibility is not set.
28	(SWC-108) State Variable Default Visibility	Low	State variable visibility is not set.
29	(SWC-108) State Variable Default Visibility	Low	State variable visibility is not set.

SushiSwapIntegration.sol



Report for contracts/yield-integrations/SushiSwapIntegration.sol
https://dashboard.mythx.io/#/console/analyses/b0abe757-d691-4c58-a1da-b024b7e00317

Line	SWC Title	Severity	Short Description
19	(SWC-123) Requirement Violation	Low	Requirement violation.
23	(SWC-108) State Variable Default Visibility	Low	State variable visibility is not set.
25	(SWC-108) State Variable Default Visibility	Low	State variable visibility is not set.
26	(SWC-108) State Variable Default Visibility	Low	State variable visibility is not set.
27	(SWC-108) State Variable Default Visibility	Low	State variable visibility is not set.
28	(SWC-108) State Variable Default Visibility	Low	State variable visibility is not set.
29	(SWC-108) State Variable Default Visibility	Low	State variable visibility is not set.

- No major issues were found by MythX.

THANK YOU FOR CHOOSING

HALBORN