HedgePie

Smart Contract Security Audit

Prepared by ShellBoxes

April 1st, 2022 - May 8th, 2022

Shellboxes.com

contact@shellboxes.com

Document Properties

Client	HedgePie
Version	1.0
Classification	Public

Scope

The HedgePie Contract in the HedgePie Repository

Repo	Commit Hash
https://github.com/innovation-upstream/hedgepie-dev/tree/dev/v1-contract	edbf2a67595085f93c8dd015e4de0681b40db040

Files	MD5 Hash
HedgepieInvestor.sol	3e0c54aa0d32204f7bfaec47e379e449
HedgepieMasterChef.sol	f5c1ae7603bc03e062a3260164785517
HedgepieStrategyManager.sol	2eaeb8635c4653c4f8dcdcab9c8b3bd1
HedgepieToken.sol	0e14af2a8e69f17e9bc8e1aceaed27a4
HedgepieYBNFT.sol	25637dcc6cbe27ea531a100cb35c1b76

Contacts

COMPANY	EMAIL
ShellBoxes	contact@shellboxes.com

Contents

1 Introduction

HedgePie engaged ShellBoxes to conduct a security assessment on the HedgePie beginning on April 1st, 2022 and ending May 8th, 2022. In this report, we detail our methodical approach to evaluate potential security issues associated with the implementation of smart contracts, by exposing possible semantic discrepancies between the smart contract code and design document, and by recommending additional ideas to optimize the existing code. Our findings indicate that the current version of smart contracts can still be enhanced further due to the presence of many security and performance concerns.

This document summarizes the findings of our audit.

1.1 About HedgePie

HedgePie is a hedge-funds based layer on top of decentralized finance that allows skilled investors and traders to design investment strategies that others can then invest into. Like the stock market, people can diversify their investment by buying into an index-style fund composed of various assets. In addition, subject matter experts can design investment strategies and publish them so others can buy into them, allowing non-experts to access well-derived strategies.

Issuer	HedgePie
Website	`https://hedgepie.com/`
Type	Solidity Smart Contract
Audit Method	Whitebox

1.2 Approach & Methodology

ShellBoxes used a combination of manual and automated security testing to achieve a balance between efficiency, timeliness, practicability, and correctness within the audit's scope. While manual testing is advised for identifying problems in logic, procedure, and implementation, automated testing techniques help to expand the coverage of smart contracts and can quickly detect code that does not comply with security best practices.

1.2.1 Risk Methodology

Vulnerabilities or bugs identified by ShellBoxes are ranked using a risk assessment technique that considers both the LIKELIHOOD and IMPACT of a security incident. This framework is effective at conveying the features and consequences of technological vulnerabilities.

Its quantitative paradigm enables repeatable and precise measurement, while also revealing the underlying susceptibility characteristics that were used to calculate the Risk scores. A risk level will be assigned to each vulnerability on a scale of 5 to 1, with 5 indicating the greatest possibility or impact.

- Likelihood quantifies the probability of a certain vulnerability being discovered and exploited in the untamed.
- Impact quantifies the technical and economic costs of a successful attack.
- Severity indicates the risk's overall criticality.

Probability and impact are classified into three categories: H, M, and L, which correspond to high, medium, and low, respectively. Severity is determined by probability and impact and is categorized into four levels, namely Critical, High, Medium, and Low.

Impact				
	High	Critical	High	Medium
	Medium	High	Medium	Low
	Low	Medium	Low	Low
		High	Medium	Low

Likelihood

2 Findings Overview

2.1 Summary

The following is a synopsis of our conclusions from our analysis of the HedgePie implementation. During the first part of our audit, we examine the smart contract source code and run the codebase via a static code analyzer. The objective here is to find known coding problems statically and then manually check (reject or confirm) issues highlighted by the tool. Additionally, we check business logics, system processes, and DeFi-related components manually to identify potential hazards and/or defects.

2.2 Key Findings

In general, these smart contracts are well-designed and constructed, but their implementation might be improved by addressing the discovered flaws, which include , 1 high-severity, 4 medium-severity, 14 low-severity vulnerabilities.

Vulnerabilities	Severity	Status
Owner Can Set Rewards To Zero	HIGH	Fixed
Owner Can Create Duplicate Pools	MEDIUM	Fixed
Reward Miscalculation	MEDIUM	Fixed
RewardDebt can be equal to 0	MEDIUM	Acknowledged
Possible Desynchronization in the userStrategyInfo	MEDIUM	Fixed
Missing Address Verification	LOW	Fixed
For Loop Over Dynamic Array	LOW	Acknowledged
Renounce Ownership	LOW	Acknowledged
Floating Pragma	LOW	Not Fixed
For Loop Over Dynamic Array	LOW	Acknowledged
Renounce Ownership	LOW	Acknowledged
Floating Pragma	LOW	Not Fixed
Missing Value Verification	LOW	Fixed
For Loop Over Dynamic Array	LOW	Acknowledged

Renounce Ownership	LOW	Acknowledged
Floating Pragma	LOW	Not Fixed
Approve Race	LOW	Acknowledged
Floating Pragma	LOW	Not Fixed
Floating Pragma	LOW	Not Fixed

3 Finding Details

A HedgepieMasterChef.sol

A.1 Owner Can Set Rewards To Zero [HIGH]

Description:

The owner can update the BONUS_MULTIPLIER to zero. This implies that the pending rewards can be always set to zero.

Code:

Listing 1: HedgepieMasterChef.sol

```
190 function updateMultiplier(uint256 _multiplierNumber) public onlyOwner {
191     BONUS_MULTIPLIER = _multiplierNumber;
192 }
```

Listing 2: HedgepieMasterChef.sol

```
214 function updatePool(uint256 _pid) public {
215     PoolInfo storage pool = poolInfo[_pid];
216     if (block.number <= pool.lastRewardBlock) {
217         return;
218     }
219     uint256 lpSupply = pool.lpToken.balanceOf(address(this));
220     if (lpSupply == 0) {
221         pool.lastRewardBlock = block.number;
222         return;
223     }
224     uint256 multiplier = getMultiplier(pool.lastRewardBlock, block.
        ↪ number);
225     uint256 hpieReward = multiplier
226         .mul(rewardPerBlock)
227         .mul(pool.allocPoint)
```

```
228             .div(totalAllocPoint);
229         pool.accHpiePerShare = pool.accHpiePerShare.add(
230             hpieReward.mul(1e12).div(lpSupply)
231         );
232         pool.lastRewardBlock = block.number;
233     }
```

Risk Level:

Likelihood – 3
Impact - 5

Recommendation:

It's recommended to add a require making sure the BONUS_MULTIPLIER cannot be set to 0.

Status - Fixed

The Hedgepie team has fixed the issue by adding a require statement to make sure that the value of the BONUS_MULTIPLIER cannot go lower than 100.

A.2 Owner Can Create Duplicate Pools [MEDIUM]

Description:

The add() function is used to add a new pool, it turns out that it did not complete essential sanity checks to prohibit the creation of a new pool with duplicate LP tokens. If a new pool with a duplicate LP token is introduced, it is likely that an error in the reward distribution to the pools and staking will occur.

Code:

Listing 3: HedgepieMasterChef.sol

```
143 function add(
144     uint256 _allocPoint,
```

```
145     IBEP20 _lpToken,
146     bool _withUpdate
147 ) public onlyOwner {
148     if (_withUpdate) {
149         massUpdatePools();
150     }
151     totalAllocPoint = totalAllocPoint.add(_allocPoint);
152     poolInfo.push(
153         PoolInfo({
154             lpToken: _lpToken,
155             allocPoint: _allocPoint,
156             lastRewardBlock: block.number,
157             accHpiePerShare: 0,
158             totalShares: 0
159         })
160     );
161 }
```

Risk Level:

Likelihood – 1
Impact - 3

Recommendation:

The risk can be remediated by defining a mapping from addresses to booleans, such that once added, LP tokens are mapped to true. A require-statement might then be added to the method to prevent the same LP token from being added once again.

Status - Fixed

Th Hedgepie team has fixed the issue by verifying that there is no existing liquidity pool with the same token before creating it.

A.3 Reward Miscalculation [MEDIUM]

Description:

The totalAllocPoint variable is used to determine the portion of total rewards minted that each pool would get, making it a critical part in the rewards calculation. As a result, if the totalAllocPoint variable is changed without first updating the pending awards, the payout for each pool is calculated improperly. The following add() and set() functions modify the totalAllocPoint variable without updating the awards.

Code:

Listing 4: HedgepieMasterChef.sol

```
143 function add(
144     uint256 _allocPoint,
145     IBEP20 _lpToken,
146     bool _withUpdate
147 ) public onlyOwner {
148     if (_withUpdate) {
149         massUpdatePools();
150     }
151     totalAllocPoint = totalAllocPoint.add(_allocPoint);
152     poolInfo.push(
153         PoolInfo({
154             lpToken: _lpToken,
155             allocPoint: _allocPoint,
156             lastRewardBlock: block.number,
157             accHpiePerShare: 0,
158             totalShares: 0
159         })
160     );
161 }
```

Risk Level:

Likelihood – 2
Impact - 4

Recommendation:

The Team should remove _withUpdate variable in the set() and add() functions and always calling the massUpdatePools() function before updating totalAllocPoint variable.

Status - Fixed

The hedgepie team has fixed the issue by removing the _withUpdate variable from the set() and add() functions.

A.4 RewardDebt can be equal to 0 [MEDIUM]

Description:

When calling the withdraw function, we are calculating the rewardDebt of the user, the issue here is that if the user.amount multiplied by the accHpiePerShare is less than 1e12 the rewardDebt will be equal to 0 thus user will not be rewarded.

Code:

Listing 5: HedgepieMasterChef.sol

```
276     user.amount = user.amount.add(_amount);
277 }
278 user.rewardDebt = user.amount.mul(pool.accHpiePerShare).div(1e12);

280 emit Deposit(msg.sender, _pid, _amount);
```

Risk Level:

Likelihood – 2
Impact - 4

Recommendation:

Ensure that the user.amount x pool.accHpiePerShare is always greater than 1e12.

Status - Acknowledged

The Hedgepie team has acknowledged the risk

A.5 Missing Address Verification [LOW]

Description:

Certain functions lack a safety check in the address, the address-type argument should include a zero-address test, otherwise, some of the contract's functionality may become inaccessible.

Code:

Listing 6: HedgepieMasterChef.sol

```
62  constructor(
63      IBEP20 _lp,
64      IBEP20 _rewardToken,
65      uint256 _rewardPerBlock,
66      address _rewardHolder
67  ) {
68      rewardToken = _rewardToken;
69      rewardPerBlock = _rewardPerBlock;

71      // staking pool
72      poolInfo.push(
73          PoolInfo({
74              lpToken: _lp,
75              allocPoint: 1000,
76              lastRewardBlock: block.number,
77              accHpiePerShare: 0,
```

```
            totalShares: 0
        })
    );

    totalAllocPoint = 1000;
    rewardHolder = _rewardHolder;
}
```

Listing 7: HedgepieMasterChef.sol

```
function add(
    uint256 _allocPoint,
    IBEP20 _lpToken,
    bool _withUpdate
) public onlyOwner {
    if (_withUpdate) {
        massUpdatePools();
    }
    totalAllocPoint = totalAllocPoint.add(_allocPoint);
    poolInfo.push(
        PoolInfo({
            lpToken: _lpToken,
            allocPoint: _allocPoint,
            lastRewardBlock: block.number,
            accHpiePerShare: 0,
            totalShares: 0
        })
    );
}
```

Risk Level:

Likelihood – 1

Impact - 3

Recommendation:

It is recommended to make sure the addresses provided in the arguments are different from the address(0).

Status - Fixed

The Hedgepie team had fixed the issue by requiring the _lpToken and _rewardToken to be different from the address(0).

A.6 For Loop Over Dynamic Array [LOW]

Description:

When smart contracts are deployed or their associated functions are invoked, the execution of these operations always consumes a certain quantity of gas, according to the amount of computation required to accomplish them. Modifying an unknown-size array that grows over time can result in a Denial-of-Service. Simply by having an excessively huge array, users can exceed the gas limit, therefore preventing the transaction from ever succeeding.

Code:

Listing 8: HedgepieMasterChef.sol

```
238 function massUpdatePools() public {
239     uint256 length = poolInfo.length;
240     for (uint256 pid = 0; pid < length; ++pid) {
241         updatePool(pid);
242     }
243 }
```

Risk Level:

Likelihood – 2
Impact - 2

Recommendation:

Avoid actions that involve looping across the entire data structure. If you really must loop over an array of unknown size, arrange for it to consume many blocs and thus multiple transactions.

Status - Acknowledged

The Hedgepie team has acknowledged the risk

A.7 Renounce Ownership [LOW]

Description:

Typically, the contract's owner is the account that deploys the contract. As a result, the owner can perform certain privileged activities. The renounceOwnership function is used in smart contracts to renounce ownership.

However, if the contract's ownership has never been transferred before renouncing it, it will never have an Owner, which may result in a denial of service.

Code:

Listing 9: HedgepieMasterChef.sol

```
10 contract HedgepieMasterChef is Ownable {
```

Risk Level:

Likelihood – 1
Impact - 3

Recommendation:

It is advised that the Owner cannot call renounceOwnership without first transferring ownership to a different address. Additionally, if a multi-signature wallet is utilized, executing the renounceOwnership method will require two or more users to sign the transaction. Alternatively, the Renounce Ownership functionality can be disabled by overriding it.

Status - Acknowledged

The Hedgepie team has acknowledged the risk, stating that they will transfer the ownership to a multisig wallet after deployment.

A.8 Floating Pragma [LOW]

Description:

The contract makes use of the floating-point pragma 0.8.4. Contracts should be deployed using the same compiler version and flags that were used during the testing process. Locking the pragma helps ensure that contracts are not unintentionally deployed using another pragma, such as an obsolete version, that may introduce issues in the contract system.

Code:

Listing 10: HedgepieMasterChef.sol

```
1 // SPDX-License-Identifier: AGPL-3.0-or-later
2 pragma solidity ^0.8.4;
```

Risk Level:

Likelihood – 2
Impact - 2

Recommendation:

Consider locking the pragma version. It is advised that floating pragma should not be used in production. Both truffle-config.js and hardhat.config.js support locking the pragma version.

Status - Not Fixed

The Hedgepie team has acknowledged the risk, stating that the pragma version is defined in the hardhat configuration.

B HedgepieInvestor.sol

B.1 For Loop Over Dynamic Array [LOW]

Description:

When smart contracts are deployed or their associated functions are invoked, the execution of these operations always consumes a certain quantity of gas, according to the amount of computation required to accomplish them. Modifying an unknown-size array that grows over time can result in a Denial-of-Service. Simply by having an excessively huge array, users can exceed the gas limit, therefore preventing the transaction from ever succeeding.

Code:

Listing 11: HedgepieInvestor.sol

```
89  for (uint8 idx = 0; idx < info.length; idx++) {
90      IYBNFT.Strategy memory infoItem = info[idx];

92      // swapping
93      uint256 amountIn = (_amount * infoItem.percent) / 1e4;
94      uint256 amountOut = _swapOnPCS(
95          amountIn,
96          _token,
97          infoItem.swapToken
98      );
```

Listing 12: HedgepieInvestor.sol

```
137 for (uint8 idx = 0; idx < info.length; idx++) {
138     IYBNFT.Strategy memory infoItem = info[idx];

140     // get the amount of strategy token to be withdrawn from strategy
141     uint256[] memory amounts = IPancakeRouter(infoItem.strategyAddress)
142         .getAmountsIn(
143             (_amount * infoItem.percent) / 1e4,
144             _getPaths(infoItem.swapToken, _token)
```

```
145         );

147     // unstaking into strategy
148     IStrategy(infoItem.strategyAddress).withdraw(amounts[0]);
149     userStrategyInfo[_user][infoItem.strategyAddress] -= amounts[0];

151     // swapping
152     IBEP20(infoItem.swapToken).safeApprove(swapRouter, amounts[0]);
153     amountOut += _swapOnPCS(amounts[0], infoItem.swapToken, _token);
154 }
```

Risk Level:

Likelihood – 2
Impact - 2

Recommendation:

Avoid actions that involve looping across the entire data structure. If you really must loop over an array of unknown size, arrange for it to consume many blocs and thus multiple transactions.

Status - Acknowledged

The Hedgepie team has acknowledged the risk.

B.2 Renounce Ownership [LOW]

Description:

Typically, the contract's owner is the account that deploys the contract. As a result, the owner can perform certain privileged activities. The renounceOwnership function is used in smart contracts to renounce ownership. However, if the contract's ownership has never been transferred before renouncing it, it will never have an Owner, which may result in a denial of service.

Code:

Listing 13: HedgepieInvestor.sol

```
12 contract HedgepieMasterChef is Ownable {
```

Risk Level:

Likelihood – 1
Impact - 3

Recommendation:

It is advised that the Owner cannot call renounceOwnership without first transferring ownership to a different address. Additionally, if a multi-signature wallet is utilized, executing the renounceOwnership method will require two or more users to sign the transaction. Alternatively, the Renounce Ownership functionality can be disabled by overriding it.

Status - Acknowledged

The Hedgepie team has acknowledged the risk, stating that they will transfer the ownership to a multisig wallet after deployment.

B.3 Floating Pragma [LOW]

Description:

The contract makes use of the floating-point pragma 0.8.4. Contracts should be deployed using the same compiler version and flags that were used during the testing process. Locking the pragma helps ensure that contracts are not unintentionally deployed using another pragma, such as an obsolete version, that may introduce issues in the contract system.

Code:

Listing 14: HedgepieInvestor.sol

```
1 // SPDX-License-Identifier: AGPL-3.0-or-later
```

```
2 pragma solidity ^0.8.4;
```

Risk Level:

Likelihood – 2
Impact - 2

Recommendation:

Consider locking the pragma version. It is advised that floating pragma should not be used in production. Both truffle-config.js and hardhat.config.js support locking the pragma version.

Status - Not Fixed

The Hedgepie team has acknowledged the risk, stating that the pragma version is defined in the hardhat configuration.

C HedgepieYBNFT.sol

C.1 Missing Value Verification [LOW]

Description:

Certain functions lack a safety check in the values, the values of the arguments should be verified to allow only the ones that go with the contract's logic.

Code:

Listing 15: HedgepieYBNFT.sol

```
100     _setStrategy(
101         tokenIdPointer,
102         _swapPercent,
103         _swapToken,
```

```
104             _strategyAddress
105         );
107         // set performance fee
108         performanceFee[tokenIdPointer] = _performanceFee;
110         emit Mint(address(this), tokenIdPointer);
111     }
```

Risk Level:

Likelihood – 1
Impact - 3

Recommendation:

It's recommended to verify the values provided in the arguments. The concerns can be resolved by utilizing a require statement.

Status - Fixed

The Hedgepie team has fixed the issue by verifying that the _performanceFee argument cannot exceed 1000.

C.2 For Loop Over Dynamic Array [LOW]

Description:

When smart contracts are deployed or their associated functions are invoked, the execution of these operations always consumes a certain quantity of gas, according to the amount of computation required to accomplish them. Modifying an unknown-size array that grows over time can result in a Denial-of-Service. Simply by having an excessively huge array, users can exceed the gas limit, therefore preventing the transaction from ever succeeding.

Code:

Listing 16: HedgepieYBNFT.sol

```
115 function manageToken(address[] calldata _tokens, bool _flag)
116     public
117     onlyOwner
118 {
119     for (uint8 idx = 0; idx < _tokens.length; idx++) {
120         allowedToken[_tokens[idx]] = _flag;
121     }
122 }
```

Listing 17: HedgepieYBNFT.sol

```
156 function _setStrategy(
157     uint256 _tokenId,
158     uint256[] calldata _swapPercent,
159     address[] calldata _swapToken,
160     address[] calldata _strategyAddress
161 ) internal {
162     for (uint8 idx = 0; idx < _swapToken.length; idx++) {
163         nftStrategy[_tokenId].push(
164             Strategy({
165                 percent: _swapPercent[idx],
166                 swapToken: _swapToken[idx],
167                 strategyAddress: _strategyAddress[idx]
168             })
169         );
170     }
171 }
```

Risk Level:

Likelihood – 2

Impact - 2

Recommendation:

Avoid actions that involve looping across the entire data structure. If you really must loop over an array of unknown size, arrange for it to consume many blocs and thus multiple transactions.

Status - Acknowledged

The Hedgepie team has acknowledged the risk

C.3 Renounce Ownership [LOW]

Description:

Typically, the contract's owner is the account that deploys the contract. As a result, the owner can perform certain privileged activities. The renounceOwnership function is used in smart contracts to renounce ownership. However, if the contract's ownership has never been transferred before renouncing it, it will never have an Owner, which may result in a denial of service.

Code:

Listing 18: HedgepieYBNFT.sol

```
10 contract YBNFT is BEP721, IYBNFT, Ownable {
```

Risk Level:

Likelihood – 1
Impact - 3

Recommendation:

It is advised that the Owner cannot call renounceOwnership without first transferring ownership to a different address. Additionally, if a multi-signature wallet is utilized, executing the renounceOwnership method will require two or more users to sign the transaction. Alternatively, the Renounce Ownership functionality can be disabled by overriding it.

Status - Acknowledged

The Hedgepie team has acknowledged the risk, stating that they will transfer the ownership to a multisig wallet after deployment.

C.4 Floating Pragma [LOW]

Description:

The contract makes use of the floating-point pragma 0.8.4. Contracts should be deployed using the same compiler version and flags that were used during the testing process. Locking the pragma helps ensure that contracts are not unintentionally deployed using another pragma, such as an obsolete version, that may introduce issues in the contract system.

Code:

Listing 19: HedgepieYBNFT.sol

```
1 // SPDX-License-Identifier: AGPL-3.0-or-later
2 pragma solidity ^0.8.4;
```

Risk Level:

Likelihood – 2
Impact - 2

Recommendation:

Consider locking the pragma version. It is advised that floating pragma should not be used in production. Both truffle-config.js and hardhat.config.js support locking the pragma version.

Status - Not Fixed

The Hedgepie team has acknowledged the risk, stating that the pragma version is defined in the hardhat configuration.

D HedgepieToken.sol

D.1 Approve Race [LOW]

Description:

The standard BEP20 implementation contains a widely known racing condition in its approve function, wherein a spender is able to witness the token owner broadcast a transaction altering their approval and quickly sign and broadcast a transaction using transferFrom to move the current approved amount from the owner's balance to the spender. If the spender's transaction is validated before the owner's, the spender will be able to get both approval amounts of both transactions.

Code:

Listing 20: HedgepieToken.sol

```
8  contract HedgepieToken is
9      AdminAccessRoles(msg.sender),
10     BEP20("Hedgepie Token", "HPIE")
```

Risk Level:

Likelihood – 1
Impact - 3

Recommendation:

It is recommended to use the increaseAllowance() and decreaseAllowance() functions to override the approval amount instead of the approve() function.

Status - Acknowledged

The Hedgepie team has acknowledged the risk.

D.2 Floating Pragma [LOW]

Description:

The contract makes use of the floating-point pragma 0.8.4. Contracts should be deployed using the same compiler version and flags that were used during the testing process. Locking the pragma helps ensure that contracts are not unintentionally deployed using another pragma, such as an obsolete version, that may introduce issues in the contract system.

Code:

Listing 21: HedgepieToken.sol

```
1 // SPDX-License-Identifier: AGPL-3.0-or-later
2 pragma solidity ^0.8.4;
```

Risk Level:

Likelihood – 2
Impact - 2

Recommendation:

Consider locking the pragma version. It is advised that floating pragma should not be used in production. Both truffle-config.js and hardhat.config.js support locking the pragma version.

Status - Not Fixed

The Hedgepie team has acknowledged the risk, stating that the pragma version is defined in the hardhat configuration.

E HedgepieStrategyManager.sol

E.1 Possible Desynchronization in the userStrategyInfo [MEDIUM]

Description:

In the HedgepieInvestor, we have the deposit function which let the users deposit their tokens, inside the function we are calling the deposit function and incrementing the amount in the userStrategyInfo mapping. The issue here is that when calling the deposit/withdraw directly from the HedgePieStrategyManager contract, the mapping won't be updated, thus creating a desynchronization.

Code:

Listing 22: HedgepieStrategyManager.sol

```
20 function deposit(address _strategy, uint256 _amount)
21     external
22     onlyActiveStratey(_strategy)
23 {
24     require(_amount > 0, "Amount can not be 0");

26     IStrategy(_strategy).invest(_amount);
27 }

29 function withdraw(address _strategy, uint256 _amount)
```

Risk Level:

Likelihood – 2
Impact - 4

Recommendation:

The Hedgepie team has fixed the issue by adding onlyInvestor modifier to the deposit function in HedgepieStrategyManager contract.

Status - Fixed

The Hedgepie team has fixed the issue by adding onlyInvestor modifier to the deposit function in HedgepieStrategyManager contract.

E.2 Floating Pragma [LOW]

Description:

The contract makes use of the floating-point pragma 0.8.4. Contracts should be deployed using the same compiler version and flags that were used during the testing process. Locking the pragma helps ensure that contracts are not unintentionally deployed using another pragma, such as an obsolete version, that may introduce issues in the contract system.

Code:

Listing 23: HedgepieStrategyManager.sol

```
1 // SPDX-License-Identifier: AGPL-3.0-or-later
2 pragma solidity ^0.8.4;
```

Risk Level:

Likelihood – 2
Impact - 2

Recommendation:

Consider locking the pragma version. It is advised that floating pragma should not be used in production. Both truffle-config.js and hardhat.config.js support locking the pragma version.

Status - Not Fixed

The Hedgepie team has acknowledged the risk, stating that the pragma version is defined in the hardhat configuration.

4 Static Analysis (Slither)

Description:

ShellBoxes expanded the coverage of the specific contract areas using automated testing methodologies. Slither, a Solidity static analysis framework, was one of the tools used. Slither was run on all-scoped contracts in both text and binary formats. This tool can be used to test mathematical relationships between Solidity instances statically and variables that allow for the detection of errors or inconsistent usage of the contracts' APIs throughout the entire codebase.

Results:

```
HedgepieMasterChef.pendingReward(uint256,address) (contracts/
    ↪ HedgepieMasterChef.sol#111-134) performs a multiplication on the
    ↪ result of a division:
        -hpieReward = multiplier.mul(rewardPerBlock).mul(pool.allocPoint)
            ↪ .div(totalAllocPoint) (contracts/HedgepieMasterChef.sol
            ↪ #125-128)
        -accHpiePerShare = accHpiePerShare.add(hpieReward.mul(1e12).div(
            ↪ lpSupply)) (contracts/HedgepieMasterChef.sol#129-131)
HedgepieMasterChef.updatePool(uint256) (contracts/HedgepieMasterChef.sol
    ↪ #214-233) performs a multiplication on the result of a division:
        -hpieReward = multiplier.mul(rewardPerBlock).mul(pool.allocPoint)
            ↪ .div(totalAllocPoint) (contracts/HedgepieMasterChef.sol
            ↪ #225-228)
        -pool.accHpiePerShare = pool.accHpiePerShare.add(hpieReward.mul(1
            ↪ e12).div(lpSupply)) (contracts/HedgepieMasterChef.sol
            ↪ #229-231)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #divide-before-multiply

HedgepieMasterChef.updatePool(uint256) (contracts/HedgepieMasterChef.sol
    ↪ #214-233) uses a dangerous strict equality:
```

```
        - lpSupply == 0 (contracts/HedgepieMasterChef.sol#220)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #dangerous-strict-equalities

Reentrancy in HedgepieMasterChef.deposit(uint256,uint256) (contracts/
    ↪ HedgepieMasterChef.sol#250-281):
        External calls:
        - rewardToken.safeTransferFrom(rewardHolder,address(msg.sender),
            ↪ pending) (contracts/HedgepieMasterChef.sol#262-266)
        - pool.lpToken.safeTransferFrom(address(msg.sender),address(this)
            ↪ ,_amount) (contracts/HedgepieMasterChef.sol#270-274)
        State variables written after the call(s):
        - pool.totalShares += _amount (contracts/HedgepieMasterChef.sol
            ↪ #275)
        - user.amount = user.amount.add(_amount) (contracts/
            ↪ HedgepieMasterChef.sol#276)
        - user.rewardDebt = user.amount.mul(pool.accHpiePerShare).div(1
            ↪ e12) (contracts/HedgepieMasterChef.sol#278)
Reentrancy in HedgepieMasterChef.emergencyWithdraw(uint256) (contracts/
    ↪ HedgepieMasterChef.sol#320-330):
        External calls:
        - pool.lpToken.safeTransfer(address(msg.sender),user.amount) (
            ↪ contracts/HedgepieMasterChef.sol#324)
        State variables written after the call(s):
        - pool.totalShares -= user.amount (contracts/HedgepieMasterChef.
            ↪ sol#325)
        - user.amount = 0 (contracts/HedgepieMasterChef.sol#328)
        - user.rewardDebt = 0 (contracts/HedgepieMasterChef.sol#329)
Reentrancy in HedgepieMasterChef.withdraw(uint256,uint256) (contracts/
    ↪ HedgepieMasterChef.sol#288-314):
        External calls:
        - rewardToken.safeTransferFrom(rewardHolder,address(msg.sender),
            ↪ pending) (contracts/HedgepieMasterChef.sol#300-304)
        State variables written after the call(s):
```

```
        - user.amount = user.amount.sub(_amount) (contracts/
            ↪ HedgepieMasterChef.sol#307)
Reentrancy in HedgepieMasterChef.withdraw(uint256,uint256) (contracts/
    ↪ HedgepieMasterChef.sol#288-314):
        External calls:
        - rewardToken.safeTransferFrom(rewardHolder,address(msg.sender),
            ↪ pending) (contracts/HedgepieMasterChef.sol#300-304)
        - pool.lpToken.safeTransfer(address(msg.sender),_amount) (
            ↪ contracts/HedgepieMasterChef.sol#308)
        State variables written after the call(s):
        - pool.totalShares -= _amount (contracts/HedgepieMasterChef.sol
            ↪ #309)
        - user.rewardDebt = user.amount.mul(pool.accHpiePerShare).div(1
            ↪ e12) (contracts/HedgepieMasterChef.sol#311)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #reentrancy-vulnerabilities-1

HedgepieMasterChef.add(uint256,IBEP20,bool) (contracts/
    ↪ HedgepieMasterChef.sol#143-161) should emit an event for:
        - totalAllocPoint = totalAllocPoint.add(_allocPoint) (contracts/
            ↪ HedgepieMasterChef.sol#151)
HedgepieMasterChef.set(uint256,uint256,bool) (contracts/
    ↪ HedgepieMasterChef.sol#169-184) should emit an event for:
        - totalAllocPoint = totalAllocPoint.sub(prevAllocPoint).add(
            ↪ _allocPoint) (contracts/HedgepieMasterChef.sol#180-182)
HedgepieMasterChef.updateMultiplier(uint256) (contracts/
    ↪ HedgepieMasterChef.sol#190-192) should emit an event for:
        - BONUS_MULTIPLIER = _multiplierNumber (contracts/
            ↪ HedgepieMasterChef.sol#191)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #missing-events-arithmetic

Ownable.constructor().msgSender (contracts/libraries/Ownable.sol#30)
    ↪ lacks a zero-check on :
```

```
            - _owner = msgSender (contracts/libraries/Ownable.sol#31)
HedgepieMasterChef.constructor(IBEP20,IBEP20,uint256,address).
    ↪ _rewardHolder (contracts/HedgepieMasterChef.sol#66) lacks a zero-
    ↪ check on :
            - rewardHolder = _rewardHolder (contracts/
                ↪ HedgepieMasterChef.sol#83)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #missing-zero-address-validation

HedgepieMasterChef.updatePool(uint256) (contracts/HedgepieMasterChef.sol
    ↪ #214-233) has external calls inside a loop: lpSupply = pool.
    ↪ lpToken.balanceOf(address(this)) (contracts/HedgepieMasterChef.
    ↪ sol#219)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ /#calls-inside-a-loop

Reentrancy in HedgepieMasterChef.deposit(uint256,uint256) (contracts/
    ↪ HedgepieMasterChef.sol#250-281):
        External calls:
        - rewardToken.safeTransferFrom(rewardHolder,address(msg.sender),
            ↪ pending) (contracts/HedgepieMasterChef.sol#262-266)
        - pool.lpToken.safeTransferFrom(address(msg.sender),address(this)
            ↪ ,_amount) (contracts/HedgepieMasterChef.sol#270-274)
        Event emitted after the call(s):
        - Deposit(msg.sender,_pid,_amount) (contracts/HedgepieMasterChef.
            ↪ sol#280)
Reentrancy in HedgepieMasterChef.emergencyWithdraw(uint256) (contracts/
    ↪ HedgepieMasterChef.sol#320-330):
        External calls:
        - pool.lpToken.safeTransfer(address(msg.sender),user.amount) (
            ↪ contracts/HedgepieMasterChef.sol#324)
        Event emitted after the call(s):
        - EmergencyWithdraw(msg.sender,_pid,user.amount) (contracts/
            ↪ HedgepieMasterChef.sol#326)
```

```
Reentrancy in HedgepieMasterChef.withdraw(uint256,uint256) (contracts/
    ↪ HedgepieMasterChef.sol#288-314):
        External calls:
        - rewardToken.safeTransferFrom(rewardHolder,address(msg.sender),
            ↪ pending) (contracts/HedgepieMasterChef.sol#300-304)
        - pool.lpToken.safeTransfer(address(msg.sender),_amount) (
            ↪ contracts/HedgepieMasterChef.sol#308)
        Event emitted after the call(s):
        - Withdraw(msg.sender,_pid,_amount) (contracts/HedgepieMasterChef
            ↪ .sol#313)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #reentrancy-vulnerabilities-3

Address.isContract(address) (contracts/libraries/Address.sol#25-36) uses
    ↪  assembly
        - INLINE ASM (contracts/libraries/Address.sol#32-34)
Address._functionCallWithValue(address,bytes,uint256,string) (contracts/
    ↪ libraries/Address.sol#151-179) uses assembly
        - INLINE ASM (contracts/libraries/Address.sol#171-174)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #assembly-usage

Address.functionCall(address,bytes) (contracts/libraries/Address.sol
    ↪ #86-91) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (contracts/
    ↪ libraries/Address.sol#118-130) is never used and should be
    ↪ removed
Address.functionCallWithValue(address,bytes,uint256,string) (contracts/
    ↪ libraries/Address.sol#138-149) is never used and should be
    ↪ removed
Address.sendValue(address,uint256) (contracts/libraries/Address.sol
    ↪ #54-66) is never used and should be removed
Context._msgData() (contracts/libraries/Context.sol#23-26) is never used
    ↪  and should be removed
```

```
SafeBEP20.safeApprove(IBEP20,address,uint256) (contracts/libraries/
    ↪ SafeBEP20.sol#42-59) is never used and should be removed
SafeBEP20.safeDecreaseAllowance(IBEP20,address,uint256) (contracts/
    ↪ libraries/SafeBEP20.sol#79-96) is never used and should be
    ↪ removed
SafeBEP20.safeIncreaseAllowance(IBEP20,address,uint256) (contracts/
    ↪ libraries/SafeBEP20.sol#61-77) is never used and should be
    ↪ removed
SafeMath.mod(uint256,uint256) (contracts/libraries/SafeMath.sol#55-57)
    ↪ is never used and should be removed
SafeMath.mod(uint256,uint256,string) (contracts/libraries/SafeMath.sol
    ↪ #59-66) is never used and should be removed
SafeMath.sqrrt(uint256) (contracts/libraries/SafeMath.sol#69-80) is
    ↪ never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #dead-code

Low level call in Address.sendValue(address,uint256) (contracts/
    ↪ libraries/Address.sol#54-66):
        - (success) = recipient.call{value: amount}() (contracts/
            ↪ libraries/Address.sol#61)
Low level call in Address._functionCallWithValue(address,bytes,uint256,
    ↪ string) (contracts/libraries/Address.sol#151-179):
        - (success,returndata) = target.call{value: weiValue}(data) (
            ↪ contracts/libraries/Address.sol#160-162)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #low-level-calls

Parameter HedgepieMasterChef.getMultiplier(uint256,uint256)._from (
    ↪ contracts/HedgepieMasterChef.sol#98) is not in mixedCase
Parameter HedgepieMasterChef.getMultiplier(uint256,uint256)._to (
    ↪ contracts/HedgepieMasterChef.sol#98) is not in mixedCase
Parameter HedgepieMasterChef.pendingReward(uint256,address)._pid (
    ↪ contracts/HedgepieMasterChef.sol#111) is not in mixedCase
```

```
Parameter HedgepieMasterChef.pendingReward(uint256,address)._user (
    ↪ contracts/HedgepieMasterChef.sol#111) is not in mixedCase
Parameter HedgepieMasterChef.add(uint256,IBEP20,bool)._allocPoint (
    ↪ contracts/HedgepieMasterChef.sol#144) is not in mixedCase
Parameter HedgepieMasterChef.add(uint256,IBEP20,bool)._lpToken (
    ↪ contracts/HedgepieMasterChef.sol#145) is not in mixedCase
Parameter HedgepieMasterChef.add(uint256,IBEP20,bool)._withUpdate (
    ↪ contracts/HedgepieMasterChef.sol#146) is not in mixedCase
Parameter HedgepieMasterChef.set(uint256,uint256,bool)._pid (contracts/
    ↪ HedgepieMasterChef.sol#170) is not in mixedCase
Parameter HedgepieMasterChef.set(uint256,uint256,bool)._allocPoint (
    ↪ contracts/HedgepieMasterChef.sol#171) is not in mixedCase
Parameter HedgepieMasterChef.set(uint256,uint256,bool)._withUpdate (
    ↪ contracts/HedgepieMasterChef.sol#172) is not in mixedCase
Parameter HedgepieMasterChef.updateMultiplier(uint256)._multiplierNumber
    ↪  (contracts/HedgepieMasterChef.sol#190) is not in mixedCase
Parameter HedgepieMasterChef.emergencyRewardWithdraw(uint256)._amount (
    ↪ contracts/HedgepieMasterChef.sol#198) is not in mixedCase
Parameter HedgepieMasterChef.updatePool(uint256)._pid (contracts/
    ↪ HedgepieMasterChef.sol#214) is not in mixedCase
Parameter HedgepieMasterChef.deposit(uint256,uint256)._pid (contracts/
    ↪ HedgepieMasterChef.sol#250) is not in mixedCase
Parameter HedgepieMasterChef.deposit(uint256,uint256)._amount (contracts
    ↪ /HedgepieMasterChef.sol#250) is not in mixedCase
Parameter HedgepieMasterChef.withdraw(uint256,uint256)._pid (contracts/
    ↪ HedgepieMasterChef.sol#288) is not in mixedCase
Parameter HedgepieMasterChef.withdraw(uint256,uint256)._amount (
    ↪ contracts/HedgepieMasterChef.sol#288) is not in mixedCase
Parameter HedgepieMasterChef.emergencyWithdraw(uint256)._pid (contracts/
    ↪ HedgepieMasterChef.sol#320) is not in mixedCase
Variable HedgepieMasterChef.BONUS_MULTIPLIER (contracts/
    ↪ HedgepieMasterChef.sol#39) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #conformance-to-solidity-naming-conventions
```

```
Redundant expression "this (contracts/libraries/Context.sol#24)"
    ↪ inContext (contracts/libraries/Context.sol#14-27)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #redundant-statements

add(uint256,IBEP20,bool) should be declared external:
        - HedgepieMasterChef.add(uint256,IBEP20,bool) (contracts/
            ↪ HedgepieMasterChef.sol#143-161)
set(uint256,uint256,bool) should be declared external:
        - HedgepieMasterChef.set(uint256,uint256,bool) (contracts/
            ↪ HedgepieMasterChef.sol#169-184)
updateMultiplier(uint256) should be declared external:
        - HedgepieMasterChef.updateMultiplier(uint256) (contracts/
            ↪ HedgepieMasterChef.sol#190-192)
emergencyRewardWithdraw(uint256) should be declared external:
        - HedgepieMasterChef.emergencyRewardWithdraw(uint256) (contracts/
            ↪ HedgepieMasterChef.sol#198-208)
deposit(uint256,uint256) should be declared external:
        - HedgepieMasterChef.deposit(uint256,uint256) (contracts/
            ↪ HedgepieMasterChef.sol#250-281)
withdraw(uint256,uint256) should be declared external:
        - HedgepieMasterChef.withdraw(uint256,uint256) (contracts/
            ↪ HedgepieMasterChef.sol#288-314)
emergencyWithdraw(uint256) should be declared external:
        - HedgepieMasterChef.emergencyWithdraw(uint256) (contracts/
            ↪ HedgepieMasterChef.sol#320-330)
owner() should be declared external:
        - Ownable.owner() (contracts/libraries/Ownable.sol#38-40)
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (contracts/libraries/Ownable.sol
            ↪ #57-60)
transferOwnership(address) should be declared external:
```

```
        - Ownable.transferOwnership(address) (contracts/libraries/Ownable
            ↪ .sol#66-68)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #public-function-that-could-be-declared-external
contracts/HedgepieMasterChef.sol analyzed (7 contracts with 78 detectors
    ↪ ), 61 result(s) found

Ownable.constructor().msgSender (contracts/libraries/Ownable.sol#30)
    ↪ lacks a zero-check on :
                - _owner = msgSender (contracts/libraries/Ownable.sol#31)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #missing-zero-address-validation

HedgepieInvestor._swapOnPCS(uint256,address,address) (contracts/
    ↪ HedgepieInvestor.sol#235-245) has external calls inside a loop:
    ↪ amounts = IPancakeRouter(swapRouter).swapExactTokensForTokens(
    ↪ _amountIn,0,path,address(this),1200) (contracts/HedgepieInvestor.
    ↪ sol#241-242)
HedgepieInvestor.deposit(address,address,uint256,address,uint256) (
    ↪ contracts/HedgepieInvestor.sol#76-113) has external calls inside
    ↪ a loop: IStrategyManager(strategyManager).deposit(infoItem.
    ↪ strategyAddress,amountOut) (contracts/HedgepieInvestor.sol
    ↪ #102-105)
HedgepieInvestor.withdraw(address,address,uint256,address,uint256) (
    ↪ contracts/HedgepieInvestor.sol#124-159) has external calls inside
    ↪  a loop: amounts = IPancakeRouter(infoItem.strategyAddress).
    ↪ getAmountsIn((_amount * infoItem.percent) / 1e4,_getPaths(
    ↪ infoItem.swapToken,_token)) (contracts/HedgepieInvestor.sol
    ↪ #141-145)
HedgepieInvestor.withdraw(address,address,uint256,address,uint256) (
    ↪ contracts/HedgepieInvestor.sol#124-159) has external calls inside
    ↪  a loop: IStrategy(infoItem.strategyAddress).withdraw(amounts[0])
    ↪  (contracts/HedgepieInvestor.sol#148)
```

```
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ /#calls-inside-a-loop

Reentrancy in HedgepieInvestor.deposit(address,address,uint256,address,
    ↪ uint256) (contracts/HedgepieInvestor.sol#76-113):
        External calls:
        - IBEP20(_token).safeTransferFrom(msg.sender,address(this),
            ↪ _amount) (contracts/HedgepieInvestor.sol#85)
        - IBEP20(_token).safeApprove(swapRouter,_amount) (contracts/
            ↪ HedgepieInvestor.sol#86)
        - info = IYBNFT(_nft).getNftStrategy(_tokenId) (contracts/
            ↪ HedgepieInvestor.sol#88)
        - amountOut = _swapOnPCS(amountIn,_token,infoItem.swapToken) (
            ↪ contracts/HedgepieInvestor.sol#94-98)
                - amounts = IPancakeRouter(swapRouter).
                    ↪ swapExactTokensForTokens(_amountIn,0,path,address(
                    ↪ this),1200) (contracts/HedgepieInvestor.sol
                    ↪ #241-242)
        - IBEP20(_token).safeApprove(infoItem.strategyAddress,_amount) (
            ↪ contracts/HedgepieInvestor.sol#101)
        - IStrategyManager(strategyManager).deposit(infoItem.
            ↪ strategyAddress,amountOut) (contracts/HedgepieInvestor.sol
            ↪ #102-105)
        State variables written after the call(s):
        - userStrategyInfo[_user][infoItem.strategyAddress] += amountOut
            ↪ (contracts/HedgepieInvestor.sol#107)
Reentrancy in HedgepieInvestor.deposit(address,address,uint256,address,
    ↪ uint256) (contracts/HedgepieInvestor.sol#76-113):
        External calls:
        - IBEP20(_token).safeTransferFrom(msg.sender,address(this),
            ↪ _amount) (contracts/HedgepieInvestor.sol#85)
        - IBEP20(_token).safeApprove(swapRouter,_amount) (contracts/
            ↪ HedgepieInvestor.sol#86)
```

```
        - info = IYBNFT(_nft).getNftStrategy(_tokenId) (contracts/
            ↪ HedgepieInvestor.sol#88)
        State variables written after the call(s):
        - userInfo[_user][_nft][_tokenId] += _amount (contracts/
            ↪ HedgepieInvestor.sol#110)
Reentrancy in HedgepieInvestor.withdraw(address,address,uint256,address,
    ↪ uint256) (contracts/HedgepieInvestor.sol#124-159):
        External calls:
        - info = IYBNFT(_nft).getNftStrategy(_tokenId) (contracts/
            ↪ HedgepieInvestor.sol#135)
        - IStrategy(infoItem.strategyAddress).withdraw(amounts[0]) (
            ↪ contracts/HedgepieInvestor.sol#148)
        State variables written after the call(s):
        - userStrategyInfo[_user][infoItem.strategyAddress] -= amounts[0]
            ↪  (contracts/HedgepieInvestor.sol#149)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #reentrancy-vulnerabilities-2

Reentrancy in HedgepieInvestor.deposit(address,address,uint256,address,
    ↪ uint256) (contracts/HedgepieInvestor.sol#76-113):
        External calls:
        - IBEP20(_token).safeTransferFrom(msg.sender,address(this),
            ↪ _amount) (contracts/HedgepieInvestor.sol#85)
        - IBEP20(_token).safeApprove(swapRouter,_amount) (contracts/
            ↪ HedgepieInvestor.sol#86)
        - info = IYBNFT(_nft).getNftStrategy(_tokenId) (contracts/
            ↪ HedgepieInvestor.sol#88)
        Event emitted after the call(s):
        - Deposit(_user,_nft,_tokenId,_amount) (contracts/
            ↪ HedgepieInvestor.sol#112)
Reentrancy in HedgepieInvestor.withdraw(address,address,uint256,address,
    ↪ uint256) (contracts/HedgepieInvestor.sol#124-159):
        External calls:
```

```
        - info = IYBNFT(_nft).getNftStrategy(_tokenId) (contracts/
            ↪ HedgepieInvestor.sol#135)
        - IBEP20(_token).safeTransfer(_user,amountOut) (contracts/
            ↪ HedgepieInvestor.sol#155)
        Event emitted after the call(s):
        - Withdraw(_user,_nft,_tokenId,_amount) (contracts/
            ↪ HedgepieInvestor.sol#158)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #reentrancy-vulnerabilities-3

Address.isContract(address) (contracts/libraries/Address.sol#25-36) uses
    ↪  assembly
        - INLINE ASM (contracts/libraries/Address.sol#32-34)
Address._functionCallWithValue(address,bytes,uint256,string) (contracts/
    ↪ libraries/Address.sol#151-179) uses assembly
        - INLINE ASM (contracts/libraries/Address.sol#171-174)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #assembly-usage

Address.functionCall(address,bytes) (contracts/libraries/Address.sol
    ↪ #86-91) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (contracts/
    ↪ libraries/Address.sol#118-130) is never used and should be
    ↪ removed
Address.functionCallWithValue(address,bytes,uint256,string) (contracts/
    ↪ libraries/Address.sol#138-149) is never used and should be
    ↪ removed
Address.sendValue(address,uint256) (contracts/libraries/Address.sol
    ↪ #54-66) is never used and should be removed
Context._msgData() (contracts/libraries/Context.sol#23-26) is never used
    ↪  and should be removed
SafeBEP20.safeDecreaseAllowance(IBEP20,address,uint256) (contracts/
    ↪ libraries/SafeBEP20.sol#79-96) is never used and should be
    ↪ removed
```

```
SafeBEP20.safeIncreaseAllowance(IBEP20,address,uint256) (contracts/
    ↪ libraries/SafeBEP20.sol#61-77) is never used and should be
    ↪ removed
SafeMath.add(uint256,uint256) (contracts/libraries/SafeMath.sol#6-11) is
    ↪  never used and should be removed
SafeMath.div(uint256,uint256) (contracts/libraries/SafeMath.sol#39-41)
    ↪ is never used and should be removed
SafeMath.div(uint256,uint256,string) (contracts/libraries/SafeMath.sol
    ↪ #43-53) is never used and should be removed
SafeMath.mod(uint256,uint256) (contracts/libraries/SafeMath.sol#55-57)
    ↪ is never used and should be removed
SafeMath.mod(uint256,uint256,string) (contracts/libraries/SafeMath.sol
    ↪ #59-66) is never used and should be removed
SafeMath.mul(uint256,uint256) (contracts/libraries/SafeMath.sol#28-37)
    ↪ is never used and should be removed
SafeMath.sqrrt(uint256) (contracts/libraries/SafeMath.sol#69-80) is
    ↪ never used and should be removed
SafeMath.sub(uint256,uint256) (contracts/libraries/SafeMath.sol#13-15)
    ↪ is never used and should be removed
SafeMath.sub(uint256,uint256,string) (contracts/libraries/SafeMath.sol
    ↪ #17-26) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #dead-code

Low level call in Address.sendValue(address,uint256) (contracts/
    ↪ libraries/Address.sol#54-66):
        - (success) = recipient.call{value: amount}() (contracts/
            ↪ libraries/Address.sol#61)
Low level call in Address._functionCallWithValue(address,bytes,uint256,
    ↪ string) (contracts/libraries/Address.sol#151-179):
        - (success,returndata) = target.call{value: weiValue}(data) (
            ↪ contracts/libraries/Address.sol#160-162)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #low-level-calls
```

```
Parameter HedgepieInvestor.deposit(address,address,uint256,address,
    ↪ uint256)._user (contracts/HedgepieInvestor.sol#77) is not in
    ↪ mixedCase
Parameter HedgepieInvestor.deposit(address,address,uint256,address,
    ↪ uint256)._nft (contracts/HedgepieInvestor.sol#78) is not in
    ↪ mixedCase
Parameter HedgepieInvestor.deposit(address,address,uint256,address,
    ↪ uint256)._tokenId (contracts/HedgepieInvestor.sol#79) is not in
    ↪ mixedCase
Parameter HedgepieInvestor.deposit(address,address,uint256,address,
    ↪ uint256)._token (contracts/HedgepieInvestor.sol#80) is not in
    ↪ mixedCase
Parameter HedgepieInvestor.deposit(address,address,uint256,address,
    ↪ uint256)._amount (contracts/HedgepieInvestor.sol#81) is not in
    ↪ mixedCase
Parameter HedgepieInvestor.withdraw(address,address,uint256,address,
    ↪ uint256)._user (contracts/HedgepieInvestor.sol#125) is not in
    ↪ mixedCase
Parameter HedgepieInvestor.withdraw(address,address,uint256,address,
    ↪ uint256)._nft (contracts/HedgepieInvestor.sol#126) is not in
    ↪ mixedCase
Parameter HedgepieInvestor.withdraw(address,address,uint256,address,
    ↪ uint256)._tokenId (contracts/HedgepieInvestor.sol#127) is not in
    ↪ mixedCase
Parameter HedgepieInvestor.withdraw(address,address,uint256,address,
    ↪ uint256)._token (contracts/HedgepieInvestor.sol#128) is not in
    ↪ mixedCase
Parameter HedgepieInvestor.withdraw(address,address,uint256,address,
    ↪ uint256)._amount (contracts/HedgepieInvestor.sol#129) is not in
    ↪ mixedCase
Parameter HedgepieInvestor.withdrawAll(address,address,uint256,address).
    ↪ _user (contracts/HedgepieInvestor.sol#169) is not in mixedCase
```

```
Parameter HedgepieInvestor.withdrawAll(address,address,uint256,address).
    ↪ _nft (contracts/HedgepieInvestor.sol#170) is not in mixedCase
Parameter HedgepieInvestor.withdrawAll(address,address,uint256,address).
    ↪ _tokenId (contracts/HedgepieInvestor.sol#171) is not in mixedCase
Parameter HedgepieInvestor.withdrawAll(address,address,uint256,address).
    ↪ _token (contracts/HedgepieInvestor.sol#172) is not in mixedCase
Parameter HedgepieInvestor.listNft(address)._nft (contracts/
    ↪ HedgepieInvestor.sol#185) is not in mixedCase
Parameter HedgepieInvestor.deListNft(address)._nft (contracts/
    ↪ HedgepieInvestor.sol#197) is not in mixedCase
Parameter HedgepieInvestor.setStrategyManager(address)._strategyManager
    ↪ (contracts/HedgepieInvestor.sol#209) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #conformance-to-solidity-naming-conventions

Redundant expression "this (contracts/libraries/Context.sol#24)"
    ↪ inContext (contracts/libraries/Context.sol#14-27)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #redundant-statements

owner() should be declared external:
        - Ownable.owner() (contracts/libraries/Ownable.sol#38-40)
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (contracts/libraries/Ownable.sol
            ↪ #57-60)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (contracts/libraries/Ownable
            ↪ .sol#66-68)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #public-function-that-could-be-declared-external
contracts/HedgepieInvestor.sol analyzed (11 contracts with 78 detectors)
    ↪ , 51 result(s) found
```

```
BEP721._checkOnBEP721Received(address,address,uint256,bytes) (contracts/
    ↪ type/BEP721.sol#470-500) ignores return value by IBEP721Receiver(
    ↪ to).onBEP721Received(_msgSender(),from,tokenId,_data) (contracts/
    ↪ type/BEP721.sol#477-496)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #unused-return

Variable 'BEP721._checkOnBEP721Received(address,address,uint256,bytes).
    ↪ retval (contracts/type/BEP721.sol#484)' in BEP721.
    ↪ _checkOnBEP721Received(address,address,uint256,bytes) (contracts/
    ↪ type/BEP721.sol#470-500) potentially used before declaration:
    ↪ retval == IBEP721Receiver.onBEP721Received.selector (contracts/
    ↪ type/BEP721.sol#485)
Variable 'BEP721._checkOnBEP721Received(address,address,uint256,bytes).
    ↪ reason (contracts/type/BEP721.sol#486)' in BEP721.
    ↪ _checkOnBEP721Received(address,address,uint256,bytes) (contracts/
    ↪ type/BEP721.sol#470-500) potentially used before declaration:
    ↪ reason.length == 0 (contracts/type/BEP721.sol#487)
Variable 'BEP721._checkOnBEP721Received(address,address,uint256,bytes).
    ↪ reason (contracts/type/BEP721.sol#486)' in BEP721.
    ↪ _checkOnBEP721Received(address,address,uint256,bytes) (contracts/
    ↪ type/BEP721.sol#470-500) potentially used before declaration:
    ↪ revert(uint256,uint256)(32 + reason,mload(uint256)(reason)) (
    ↪ contracts/type/BEP721.sol#493)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #pre-declaration-usage-of-local-variables

Reentrancy in YBNFT.mint(uint256[],address[],address[],uint256) (
    ↪ contracts/HedgepieYBNFT.sol#82-112):
        External calls:
        - _safeMint(msg.sender,tokenIdPointer) (contracts/HedgepieYBNFT.
            ↪ sol#98)
                - IBEP721Receiver(to).onBEP721Received(_msgSender(),from,
                    ↪ tokenId,_data) (contracts/type/BEP721.sol#477-496)
```

```
        State variables written after the call(s):
        - _setStrategy(tokenIdPointer,_swapPercent,_swapToken,
            ↪ _strategyAddress) (contracts/HedgepieYBNFT.sol#101-106)
                - nftStrategy[_tokenId].push(Strategy(_swapPercent[idx],
                    ↪ _swapToken[idx],_strategyAddress[idx])) (contracts/
                    ↪ HedgepieYBNFT.sol#163-169)
        - performanceFee[tokenIdPointer] = _performanceFee (contracts/
            ↪ HedgepieYBNFT.sol#109)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #reentrancy-vulnerabilities-2

Reentrancy in YBNFT.mint(uint256[],address[],address[],uint256) (
    ↪ contracts/HedgepieYBNFT.sol#82-112):
        External calls:
        - _safeMint(msg.sender,tokenIdPointer) (contracts/HedgepieYBNFT.
            ↪ sol#98)
                - IBEP721Receiver(to).onBEP721Received(_msgSender(),from,
                    ↪ tokenId,_data) (contracts/type/BEP721.sol#477-496)
        Event emitted after the call(s):
        - Mint(address(this),tokenIdPointer) (contracts/HedgepieYBNFT.sol
            ↪ #111)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #reentrancy-vulnerabilities-3

Address.isContract(address) (contracts/libraries/Address.sol#25-36) uses
    ↪  assembly
        - INLINE ASM (contracts/libraries/Address.sol#32-34)
Address._functionCallWithValue(address,bytes,uint256,string) (contracts/
    ↪ libraries/Address.sol#151-179) uses assembly
        - INLINE ASM (contracts/libraries/Address.sol#171-174)
BEP721._checkOnBEP721Received(address,address,uint256,bytes) (contracts/
    ↪ type/BEP721.sol#470-500) uses assembly
        - INLINE ASM (contracts/type/BEP721.sol#492-494)
```

```
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #assembly-usage

Address.functionCall(address,bytes) (contracts/libraries/Address.sol
    ↪ #86-91) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (contracts/
    ↪ libraries/Address.sol#118-130) is never used and should be
    ↪ removed
Address.functionCallWithValue(address,bytes,uint256,string) (contracts/
    ↪ libraries/Address.sol#138-149) is never used and should be
    ↪ removed
Address.sendValue(address,uint256) (contracts/libraries/Address.sol
    ↪ #54-66) is never used and should be removed
BEP721._burn(uint256) (contracts/type/BEP721.sol#383-397) is never used
    ↪ and should be removed
Context._msgData() (contracts/libraries/Context.sol#23-26) is never used
    ↪  and should be removed
SafeBEP20.safeApprove(IBEP20,address,uint256) (contracts/libraries/
    ↪ SafeBEP20.sol#42-59) is never used and should be removed
SafeBEP20.safeDecreaseAllowance(IBEP20,address,uint256) (contracts/
    ↪ libraries/SafeBEP20.sol#79-96) is never used and should be
    ↪ removed
SafeBEP20.safeIncreaseAllowance(IBEP20,address,uint256) (contracts/
    ↪ libraries/SafeBEP20.sol#61-77) is never used and should be
    ↪ removed
SafeBEP20.safeTransfer(IBEP20,address,uint256) (contracts/libraries/
    ↪ SafeBEP20.sol#12-21) is never used and should be removed
SafeMath.add(uint256,uint256) (contracts/libraries/SafeMath.sol#6-11) is
    ↪  never used and should be removed
SafeMath.div(uint256,uint256) (contracts/libraries/SafeMath.sol#39-41)
    ↪ is never used and should be removed
SafeMath.div(uint256,uint256,string) (contracts/libraries/SafeMath.sol
    ↪ #43-53) is never used and should be removed
```

```
SafeMath.mod(uint256,uint256) (contracts/libraries/SafeMath.sol#55-57)
    ↪ is never used and should be removed
SafeMath.mod(uint256,uint256,string) (contracts/libraries/SafeMath.sol
    ↪ #59-66) is never used and should be removed
SafeMath.mul(uint256,uint256) (contracts/libraries/SafeMath.sol#28-37)
    ↪ is never used and should be removed
SafeMath.sqrrt(uint256) (contracts/libraries/SafeMath.sol#69-80) is
    ↪ never used and should be removed
SafeMath.sub(uint256,uint256) (contracts/libraries/SafeMath.sol#13-15)
    ↪ is never used and should be removed
SafeMath.sub(uint256,uint256,string) (contracts/libraries/SafeMath.sol
    ↪ #17-26) is never used and should be removed
Strings.toHexString(uint256) (contracts/libraries/Strings.sol#35-46) is
    ↪ never used and should be removed
Strings.toHexString(uint256,uint256) (contracts/libraries/Strings.sol
    ↪ #51-65) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #dead-code

Low level call in Address.sendValue(address,uint256) (contracts/
    ↪ libraries/Address.sol#54-66):
        - (success) = recipient.call{value: amount}() (contracts/
            ↪ libraries/Address.sol#61)
Low level call in Address._functionCallWithValue(address,bytes,uint256,
    ↪ string) (contracts/libraries/Address.sol#151-179):
        - (success,returndata) = target.call{value: weiValue}(data) (
            ↪ contracts/libraries/Address.sol#160-162)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #low-level-calls

Parameter YBNFT.getNftStrategy(uint256)._tokenId (contracts/
    ↪ HedgepieYBNFT.sol#45) is not in mixedCase
Parameter YBNFT.getPerformanceFee(uint256)._tokenId (contracts/
    ↪ HedgepieYBNFT.sol#54) is not in mixedCase
```

```
Parameter YBNFT.setInvestor(address)._investor (contracts/HedgepieYBNFT.
    ↪ sol#62) is not in mixedCase
Parameter YBNFT.setLottery(address)._lottery (contracts/HedgepieYBNFT.
    ↪ sol#67) is not in mixedCase
Parameter YBNFT.setTreasury(address)._treasury (contracts/HedgepieYBNFT.
    ↪ sol#72) is not in mixedCase
Parameter YBNFT.setProtocolFee(uint256)._protocolFee (contracts/
    ↪ HedgepieYBNFT.sol#77) is not in mixedCase
Parameter YBNFT.mint(uint256[],address[],address[],uint256)._swapPercent
    ↪  (contracts/HedgepieYBNFT.sol#83) is not in mixedCase
Parameter YBNFT.mint(uint256[],address[],address[],uint256)._swapToken (
    ↪ contracts/HedgepieYBNFT.sol#84) is not in mixedCase
Parameter YBNFT.mint(uint256[],address[],address[],uint256).
    ↪ _strategyAddress (contracts/HedgepieYBNFT.sol#85) is not in
    ↪ mixedCase
Parameter YBNFT.mint(uint256[],address[],address[],uint256).
    ↪ _performanceFee (contracts/HedgepieYBNFT.sol#86) is not in
    ↪ mixedCase
Parameter YBNFT.manageToken(address[],bool)._tokens (contracts/
    ↪ HedgepieYBNFT.sol#115) is not in mixedCase
Parameter YBNFT.manageToken(address[],bool)._flag (contracts/
    ↪ HedgepieYBNFT.sol#115) is not in mixedCase
Parameter YBNFT.deposit(uint256,address,uint256)._tokenId (contracts/
    ↪ HedgepieYBNFT.sol#125) is not in mixedCase
Parameter YBNFT.deposit(uint256,address,uint256)._token (contracts/
    ↪ HedgepieYBNFT.sol#126) is not in mixedCase
Parameter YBNFT.deposit(uint256,address,uint256)._amount (contracts/
    ↪ HedgepieYBNFT.sol#127) is not in mixedCase
Parameter YBNFT.withdraw(uint256,address,uint256)._tokenId (contracts/
    ↪ HedgepieYBNFT.sol#136) is not in mixedCase
Parameter YBNFT.withdraw(uint256,address,uint256)._token (contracts/
    ↪ HedgepieYBNFT.sol#137) is not in mixedCase
Parameter YBNFT.withdraw(uint256,address,uint256)._amount (contracts/
    ↪ HedgepieYBNFT.sol#138) is not in mixedCase
```

```
Parameter YBNFT.withdraw(uint256,address)._tokenId (contracts/
    ↪ HedgepieYBNFT.sol#146) is not in mixedCase
Parameter YBNFT.withdraw(uint256,address)._token (contracts/
    ↪ HedgepieYBNFT.sol#146) is not in mixedCase
Parameter BEP721.safeTransferFrom(address,address,uint256,bytes)._data (
    ↪ contracts/type/BEP721.sol#230) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #conformance-to-solidity-naming-conventions

Redundant expression "this (contracts/libraries/Context.sol#24)"
    ↪ inContext (contracts/libraries/Context.sol#14-27)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #redundant-statements

manageToken(address[],bool) should be declared external:
        - YBNFT.manageToken(address[],bool) (contracts/HedgepieYBNFT.sol
            ↪ #115-122)
owner() should be declared external:
        - Ownable.owner() (contracts/libraries/Ownable.sol#38-40)
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (contracts/libraries/Ownable.sol
            ↪ #57-60)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (contracts/libraries/Ownable
            ↪ .sol#66-68)
balanceOf(address) should be declared external:
        - BEP721.balanceOf(address) (contracts/type/BEP721.sol#61-73)
name() should be declared external:
        - BEP721.name() (contracts/type/BEP721.sol#96-98)
symbol() should be declared external:
        - BEP721.symbol() (contracts/type/BEP721.sol#103-105)
tokenURI(uint256) should be declared external:
        - BEP721.tokenURI(uint256) (contracts/type/BEP721.sol#110-127)
approve(address,uint256) should be declared external:
```

```
        - BEP721.approve(address,uint256) (contracts/type/BEP721.sol
            ↪ #141-151)
setApprovalForAll(address,bool) should be declared external:
        - BEP721.setApprovalForAll(address,bool) (contracts/type/BEP721.
            ↪ sol#174-180)
transferFrom(address,address,uint256) should be declared external:
        - BEP721.transferFrom(address,address,uint256) (contracts/type/
            ↪ BEP721.sol#198-210)
safeTransferFrom(address,address,uint256) should be declared external:
        - BEP721.safeTransferFrom(address,address,uint256) (contracts/
            ↪ type/BEP721.sol#215-221)
onBEP721Received(address,address,uint256,bytes) should be declared
    ↪ external:
        - BEP721.onBEP721Received(address,address,uint256,bytes) (
            ↪ contracts/type/BEP721.sol#242-249)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #public-function-that-could-be-declared-external
contracts/HedgepieYBNFT.sol analyzed (17 contracts with 78 detectors),
    ↪ 67 result(s) found

BEP20.allowance(address,address).owner (contracts/type/BEP20.sol#125)
    ↪ shadows:
        - Ownable.owner() (contracts/libraries/Ownable.sol#38-40) (
            ↪ function)
BEP20._approve(address,address,uint256).owner (contracts/type/BEP20.sol
    ↪ #332) shadows:
        - Ownable.owner() (contracts/libraries/Ownable.sol#38-40) (
            ↪ function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #local-variable-shadowing

Ownable.constructor().msgSender (contracts/libraries/Ownable.sol#30)
    ↪ lacks a zero-check on :
                - _owner = msgSender (contracts/libraries/Ownable.sol#31)
```

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

Address.isContract(address) (contracts/libraries/Address.sol#25-36) uses
    ↪  assembly
        - INLINE ASM (contracts/libraries/Address.sol#32-34)
Address._functionCallWithValue(address,bytes,uint256,string) (contracts/
    ↪ libraries/Address.sol#151-179) uses assembly
        - INLINE ASM (contracts/libraries/Address.sol#171-174)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #assembly-usage

Address._functionCallWithValue(address,bytes,uint256,string) (contracts/
    ↪ libraries/Address.sol#151-179) is never used and should be
    ↪ removed
Address.functionCall(address,bytes) (contracts/libraries/Address.sol
    ↪ #86-91) is never used and should be removed
Address.functionCall(address,bytes,string) (contracts/libraries/Address.
    ↪ sol#99-105) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (contracts/
    ↪ libraries/Address.sol#118-130) is never used and should be
    ↪ removed
Address.functionCallWithValue(address,bytes,uint256,string) (contracts/
    ↪ libraries/Address.sol#138-149) is never used and should be
    ↪ removed
Address.isContract(address) (contracts/libraries/Address.sol#25-36) is
    ↪ never used and should be removed
Address.sendValue(address,uint256) (contracts/libraries/Address.sol
    ↪ #54-66) is never used and should be removed
BEP20._burn(address,uint256) (contracts/type/BEP20.sol#306-316) is never
    ↪  used and should be removed
BEP20._burnFrom(address,uint256) (contracts/type/BEP20.sol#349-359) is
    ↪ never used and should be removed
```

```
Context._msgData() (contracts/libraries/Context.sol#23-26) is never used
    ↪  and should be removed
EnumerableSet.add(EnumerableSet.UintSet,uint256) (contracts/libraries/
    ↪ EnumerableSet.sol#229-231) is never used and should be removed
EnumerableSet.at(EnumerableSet.UintSet,uint256) (contracts/libraries/
    ↪ EnumerableSet.sol#274-280) is never used and should be removed
EnumerableSet.contains(EnumerableSet.UintSet,uint256) (contracts/
    ↪ libraries/EnumerableSet.sol#249-255) is never used and should be
    ↪ removed
EnumerableSet.length(EnumerableSet.UintSet) (contracts/libraries/
    ↪ EnumerableSet.sol#260-262) is never used and should be removed
EnumerableSet.remove(EnumerableSet.UintSet,uint256) (contracts/libraries
    ↪ /EnumerableSet.sol#239-244) is never used and should be removed
SafeMath.div(uint256,uint256) (contracts/libraries/SafeMath.sol#39-41)
    ↪ is never used and should be removed
SafeMath.div(uint256,uint256,string) (contracts/libraries/SafeMath.sol
    ↪ #43-53) is never used and should be removed
SafeMath.mod(uint256,uint256) (contracts/libraries/SafeMath.sol#55-57)
    ↪ is never used and should be removed
SafeMath.mod(uint256,uint256,string) (contracts/libraries/SafeMath.sol
    ↪ #59-66) is never used and should be removed
SafeMath.mul(uint256,uint256) (contracts/libraries/SafeMath.sol#28-37)
    ↪ is never used and should be removed
SafeMath.sqrrt(uint256) (contracts/libraries/SafeMath.sol#69-80) is
    ↪ never used and should be removed
SafeMath.sub(uint256,uint256) (contracts/libraries/SafeMath.sol#13-15)
    ↪ is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #dead-code

Low level call in Address.sendValue(address,uint256) (contracts/
    ↪ libraries/Address.sol#54-66):
        - (success) = recipient.call{value: amount}() (contracts/
            ↪ libraries/Address.sol#61)
```

```
Low level call in Address._functionCallWithValue(address,bytes,uint256,
    ↪ string) (contracts/libraries/Address.sol#151-179):
        - (success,returndata) = target.call{value: weiValue}(data) (
            ↪ contracts/libraries/Address.sol#160-162)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #low-level-calls

Parameter HedgepieToken.mint(address,uint256)._to (contracts/
    ↪ HedgepieToken.sol#21) is not in mixedCase
Parameter HedgepieToken.mint(address,uint256)._amount (contracts/
    ↪ HedgepieToken.sol#21) is not in mixedCase
Parameter EnumerableSet.addrToUint(address)._address (contracts/
    ↪ libraries/EnumerableSet.sol#151) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #conformance-to-solidity-naming-conventions

Redundant expression "this (contracts/libraries/Context.sol#24)"
    ↪ inContext (contracts/libraries/Context.sol#14-27)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #redundant-statements

renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (contracts/libraries/Ownable.sol
            ↪ #57-60)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (contracts/libraries/Ownable
            ↪ .sol#66-68)
getRoleMemberCount(bytes32) should be declared external:
        - AccessControl.getRoleMemberCount(bytes32) (contracts/type/
            ↪ AccessControl.sol#106-108)
getRoleMember(bytes32,uint256) should be declared external:
        - AccessControl.getRoleMember(bytes32,uint256) (contracts/type/
            ↪ AccessControl.sol#122-128)
getRoleAdmin(bytes32) should be declared external:
```

```
        - AccessControl.getRoleAdmin(bytes32) (contracts/type/
            ↪ AccessControl.sol#136-138)
addMintUser(address) should be declared external:
        - AdminAccessRoles.addMintUser(address) (contracts/type/
            ↪ AdminAccessRoles.sol#41-43)
addAdmin(address) should be declared external:
        - AdminAccessRoles.addAdmin(address) (contracts/type/
            ↪ AdminAccessRoles.sol#46-48)
removeMintUser(address) should be declared external:
        - AdminAccessRoles.removeMintUser(address) (contracts/type/
            ↪ AdminAccessRoles.sol#51-53)
renounceAdmin() should be declared external:
        - AdminAccessRoles.renounceAdmin() (contracts/type/
            ↪ AdminAccessRoles.sol#56-58)
name() should be declared external:
        - BEP20.name() (contracts/type/BEP20.sol#73-75)
decimals() should be declared external:
        - BEP20.decimals() (contracts/type/BEP20.sol#80-82)
symbol() should be declared external:
        - BEP20.symbol() (contracts/type/BEP20.sol#87-89)
balanceOf(address) should be declared external:
        - BEP20.balanceOf(address) (contracts/type/BEP20.sol#101-103)
transfer(address,uint256) should be declared external:
        - BEP20.transfer(address,uint256) (contracts/type/BEP20.sol
            ↪ #113-120)
allowance(address,address) should be declared external:
        - BEP20.allowance(address,address) (contracts/type/BEP20.sol
            ↪ #125-132)
approve(address,uint256) should be declared external:
        - BEP20.approve(address,uint256) (contracts/type/BEP20.sol
            ↪ #141-148)
transferFrom(address,address,uint256) should be declared external:
        - BEP20.transferFrom(address,address,uint256) (contracts/type/
            ↪ BEP20.sol#162-177)
```

```
increaseAllowance(address,uint256) should be declared external:
        - BEP20.increaseAllowance(address,uint256) (contracts/type/BEP20.
          ↪ sol#191-201)
decreaseAllowance(address,uint256) should be declared external:
        - BEP20.decreaseAllowance(address,uint256) (contracts/type/BEP20.
          ↪ sol#217-230)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #public-function-that-could-be-declared-external
contracts/HedgepieToken.sol analyzed (10 contracts with 78 detectors),
    ↪ 52 result(s) found


Context._msgData() (contracts/libraries/Context.sol#23-26) is never used
    ↪  and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #dead-code

Parameter HedgepieStrategyManager.deposit(address,uint256)._strategy (
    ↪ contracts/HedgepieStrategyManager.sol#20) is not in mixedCase
Parameter HedgepieStrategyManager.deposit(address,uint256)._amount (
    ↪ contracts/HedgepieStrategyManager.sol#20) is not in mixedCase
Parameter HedgepieStrategyManager.withdraw(address,uint256)._strategy (
    ↪ contracts/HedgepieStrategyManager.sol#29) is not in mixedCase
Parameter HedgepieStrategyManager.withdraw(address,uint256)._amount (
    ↪ contracts/HedgepieStrategyManager.sol#29) is not in mixedCase
Parameter HedgepieStrategyManager.addStrategy(address)._strategy (
    ↪ contracts/HedgepieStrategyManager.sol#43) is not in mixedCase
Parameter HedgepieStrategyManager.removeStrategy(address)._strategy (
    ↪ contracts/HedgepieStrategyManager.sol#55) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #conformance-to-solidity-naming-conventions

Redundant expression "this (contracts/libraries/Context.sol#24)"
    ↪ inContext (contracts/libraries/Context.sol#14-27)
```

```
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #redundant-statements

owner() should be declared external:
        - Ownable.owner() (contracts/libraries/Ownable.sol#38-40)
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (contracts/libraries/Ownable.sol
            ↪ #57-60)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (contracts/libraries/Ownable
            ↪ .sol#66-68)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #public-function-that-could-be-declared-external
contracts/HedgepieStrategyManager.sol analyzed (4 contracts with 78
    ↪ detectors), 11 result(s) found
```

Conclusion:

Most of the vulnerabilities found by the analysis have already been addressed by the smart contract code review.

5 Conclusion

In this audit, we examined the design and implementation of HedgePie contract and discovered several issues of varying severity. HedgePie team addressed 6 issues raised in the initial report and implemented the necessary fixes, while classifying the rest as a risk with low-probability of occurrence. Shellboxes' auditors advised HedgePie Team to maintain a high level of vigilance and to keep those findings in mind in order to avoid any future complications.

For a Contract Audit, contact us at contact@shellboxes.com