Velvet Capital

Smart Contract Security Audit

Prepared by ShellBoxes

August 26th, 2022 - September 12th, 2022

Shellboxes.com

contact@shellboxes.com

Document Properties

Client	Velvet Capital
Version	1.0
Classification	Public

Scope

The Velvet Capital Contract in the Velvet Capital Repository

Repo	Commit Hash
https://github.com/Velvet-Capital/protocols	ba5b6b32df759f630e325fe79d7ea2f5b27adbcc

Files	MD5 Hash
IndexFactory.sol	90b903abbee8d3408911e68590eb78d9
venus/TokenMetadata.sol	4f29f2a716c50bc79a9985796ad71016
vault/Vault.sol	41dc52219f88b7b837049292499865b1
vault/VelvetSafeModule.sol	dc36378af5ebdc1b1c05fd0afcf574ab
rebalance/Rebalancing.sol	3065c03665f7ad41cad87df676ed03d6
oracle/PriceOracle.sol	d40d2841203eb00bb0f89ef49fe48e0c
core/Adapter.sol	c3320d2591ecf6fd77aa69b34b134904
core/IndexSwap.sol	c7b32711672a5aa40e20071ef8865478
core/IndexSwapLibrary.sol	5cf0e02bbc0fab372840bb8da87bc523
access/AccessController.sol	7c6ebf3f7895b68592678d8ae60044d6

Re-Audit Scope

Repo	Commit Hash
https://github.com/Velvet-Capital/protocols	b1fc3df8865aa4c457fe4786791309e404bc6478

Files	MD5 Hash
IndexFactory.sol	bff7883ec6ed23ff2d1fa5a81e8d05ad
vault/Vault.sol	a904e54d8d8f940fb9a4fc5acd09cf43
vault/VelvetSafeModule.sol	0e1591a95eee0c75a23a3b23528a4b47
rebalance/Rebalancing.sol	eb49c73a9e3a6875f7267af24d2274e4
oracle/PriceOracle.sol	3aa3a42d284315f26e8e41d81ac8e0a8
core/Adapter.sol	f8703ee74037b3b2595e43a77ef0c3b8
core/IndexSwap.sol	79f7f2beaea4c21c3bb0a4a416fc39d1
core/IndexSwapLibrary.sol	f2c944c2350b6ab9369e4c3ea1bf3fa1
access/AccessController.sol	c5d6636302b07cb401ac2d7d8f55b8a5

Contacts

COMPANY	EMAIL
ShellBoxes	contact@shellboxes.com

Contents

1 Introduction

Velvet Capital engaged ShellBoxes to conduct a security assessment on the Velvet Capital beginning on August 26th, 2022 and ending September 12th, 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 Velvet Capital

Velvet Capital is a DeFi protocol that helps people & institutions create tokenized index funds, portfolios & other financial products with additional yield.

The protocol provides all the necessary infrastructure for financial product development being integrated with AMMs, Lending protocols and other DeFi primitives to give users a diverse asset management toolkit.

Issuer	Velvet Capital
Website	`https://velvet.capital`
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 Velvet Capital 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 , 2 high-severity, 1 medium-severity, 11 low-severity, 10 informational-severity vulnerabilities.

Vulnerabilities	Severity	Status
A.1. The Investor Can Spend His Funds Without Minting Index Tokens	HIGH	Fixed
A.2. The User's tokens Can Get Burned Without Withdrawing All His Funds	HIGH	Fixed
A.3. The Record Index Can Overflow	MEDIUM	Fixed
A.4. Missing Value Verification	LOW	Fixed
A.5. Missing Address Verification	LOW	Fixed
A.6. Owner Can Renounce Ownership	LOW	Acknowledged
B.1. Missing Address Verification	LOW	Fixed
B.2. Owner Can Renounce Ownership	LOW	Acknowledged
C.1. Owner Can Renounce Ownership	LOW	Acknowledged
D.1. Missing Address Verification	LOW	Fixed
E.1. Missing Address Verification	LOW	Fixed
F.1. Owner Can Renounce Ownership	LOW	Acknowledged

G.1. Missing Address Verification	LOW	Fixed
I.1. Missing Address Verification	LOW	Fixed
A.7. Floating Pragma	INFORMATIONAL	Fixed
B.3. Floating Pragma	INFORMATIONAL	Fixed
C.2. Floating Pragma	INFORMATIONAL	Fixed
D.2. Floating Pragma	INFORMATIONAL	Fixed
E.2. Floating Pragma	INFORMATIONAL	Fixed
F.2. Floating Pragma	INFORMATIONAL	Fixed
G.2. Floating Pragma	INFORMATIONAL	Fixed
H.1. Floating Pragma	INFORMATIONAL	Fixed
I.2. Floating Pragma	INFORMATIONAL	Fixed
J.1. Floating Pragma	INFORMATIONAL	Fixed

3 Finding Details

A IndexSwap.sol

A.1 The Investor Can Spend His Funds Without Minting Index Tokens [HIGH]

Description:

The users call the investInFund function in order to invest BNB and mint an amount of index tokens. If a user invests an amount that is lower than vaultBalance/tokenBalanceInUSD[i] for every token, the investedAmountAfterSlippageBNB will be equal to zero due to a type conversion error. Therefore, the user will spend his funds without minting any index tokens that will allow him to withdraw his funds later.

Code:

Listing 1: IndexSwap.sol

```
187 amount = indexSwapLibrary.calculateSwapAmounts(
188     IIndexSwap(address(this)),
189     tokenAmount,
190     tokenBalanceInBNB,
191     vaultBalance
192 );

194 investedAmountAfterSlippage = _swapETHToTokens(
195     tokenAmount,
196     amount,
197     _slippage
198 );

200 uint256 investedAmountAfterSlippageBNB = indexSwapLibrary
201     ._getTokenPriceUSDETH(investedAmountAfterSlippage);
```

```
uint256 vaultBalanceBNB = indexSwapLibrary._getTokenPriceUSDETH(
    vaultBalance
);



if (totalSupply() > 0) {
    tokenAmount = _mintShareAmount(
        investedAmountAfterSlippageBNB,
        vaultBalanceBNB
    );
} else {
    tokenAmount = investedAmountAfterSlippageBNB;
}

_mint(msg.sender, tokenAmount);
```

Listing 2: IndexSwapLibrary.sol

```
function calculateSwapAmounts(
    IIndexSwap _index,
    uint256 tokenAmount,
    uint256[] memory tokenBalanceInUSD,
    uint256 vaultBalance
) public view returns (uint256[] memory) {
    uint256[] memory amount = new uint256[](_index.getTokens().length);
    if (_index.totalSupply() > 0) {
        for (uint256 i = 0; i < _index.getTokens().length; i++) {
            amount[i] = tokenBalanceInUSD[i].mul(tokenAmount).div(
                vaultBalance
            );
        }
    }
    return amount;
}
```

Risk Level:

Likelihood – 4
Impact - 5

Recommendation:

Consider requiring msg.value to be greater than vaultBalance/tokenBalanceInUSD[i].

Status - Fixed

The Velvet team has fixed the issue by verifying tokenBalanceInUSD[i]*tokenAmount to be greater than vaultBalance.

A.2 The User's tokens Can Get Burned Without Withdrawing All His Funds [HIGH]

Description:

The withdrawFund function is used to withdraw the funds invested in the tokens listed in the tokens array. However, there is a scenario where the amount to be withdrawn of a specific token or multiple ones can round to zero. If the tokenBalance * tokenAmount is lower than totalSupplyIndex, then the amount variable will be equal to zero due to a type conversion error.

Code:

Listing 3: IndexSwap.sol

```
233 function withdrawFund(uint256 tokenAmount, uint256 _slippage, bool
        ↪ isMultiAsset)
234     public
235     nonReentrant
236 {
237     require(!paused, "The contract is paused !");
238     require(
```

```
239         tokenAmount <= balanceOf(msg.sender),
240         "caller is not holding given token amount"
241     );

243     uint256 sumBalance=0;
244     uint256[] memory arrayOfTokenAmount = new uint256[](_tokens.length);
245     uint256 totalSupplyIndex = totalSupply();

247     _burn(msg.sender, tokenAmount);

249     for (uint256 i = 0; i < _tokens.length; i++) {
250         uint256 tokenBalance = indexSwapLibrary.getTokenBalance(
251             IIndexSwap(address(this)),
252             _tokens[i],
253             adapter.getETH() == _tokens[i]
254         );

256         uint256 amount = tokenBalance.mul(tokenAmount).div(
257             totalSupplyIndex
258         );
```

Risk Level:

Likelihood – 3

Impact - 5

Recommendation:

Given that the caller chooses the tokenAmount argument and that the worst-case scenario is 1, consider requiring the tokenBalance variable to be higher than the totalSupplyIndex variable.

Status - Fixed

The Velvet team has fixed the issue by verifying tokenBalance*tokenAmount to be greater than totalSupplyIndex.

A.3 The Record Index Can Overflow [MEDIUM]

Description:

The index attribute in the Record struct is an uint8. In the case where the length of the tokens array is greater than 255, the index attribute will overflow and start again from zero.

Code:

Listing 4: IndexSwap.sol

```
122 function initToken(address[] calldata tokens, uint96[] calldata denorms)
123     external
124     onlyOwner
125 {
126     require(tokens.length == denorms.length, "INVALID_INIT_INPUT");
127     require(_tokens.length == 0, "INITIALIZED");
128     uint256 len = tokens.length;
129     uint256 totalWeight = 0;
130     for (uint256 i = 0; i < len; i++) {
131         _records[tokens[i]] = Record({
132             lastDenormUpdate: uint40(block.timestamp),
133             denorm: denorms[i],
134             index: uint8(i)
135         });
136         _tokens.push(tokens[i]);

138         totalWeight = totalWeight.add(denorms[i]);
139     }
140     require(totalWeight == TOTAL_WEIGHT, "INVALID_WEIGHTS");
```

```
142     emit LOG_PUBLIC_SWAP_ENABLED();
143 }
```

Risk Level:

Likelihood – 3
Impact - 3

Recommendation:

Consider limiting the length of the tokens array to 255, or change the type of the index.

Status - Fixed

The Velvet team has fixed the issue by changing the type of the index attribute to uint256.

A.4 Missing Value Verification [LOW]

Description:

Certain functions lack a value safety check, the values of the arguments should be verified to allow only the ones that comply with the contract's logic. In the constructor, the contract must ensure that _maxInvestmentAmount and _feePointBasis are different from zero, also, in the initToken the contract must ensure that denorms are lower than TOTAL_WEIGHT.

Code:

Listing 5: IndexSwap.sol

```
88  constructor(
89      string memory _name,
90      string memory _symbol,
91      address _outAsset,
92      address _vault,
93      uint256 _maxInvestmentAmount,
94      address _indexSwapLibrary,
```

```
95      address _adapter,
96      address _accessController,
97      address _tokenMetadata,
98      uint256 _feePointBasis,
99      address _treasury
100 ) TokenBase(_name, _symbol) {
101     vault = _vault;
102     outAsset = _outAsset; //As now we are tacking busd
103     MAX_INVESTMENTAMOUNT = _maxInvestmentAmount;
104     indexSwapLibrary = IIndexSwapLibrary(_indexSwapLibrary);
105     adapter = IAdapter(_adapter);
106     accessController = IAccessController(_accessController);
107     tokenMetadata = TokenMetadata(_tokenMetadata);
108     paused = false;

110     feePointBasis = _feePointBasis;
111     treasury = payable(_treasury);
112 }
```

Risk Level:

Likelihood – 1
Impact - 3

Recommendation:

We recommend that you verify the values provided in the arguments. The issue can be addressed by utilizing a require statement.

Status - Fixed

The Velvet team has fixed the issue by verifying the values provided in the arguments to comply with the logic of the contract.

A.5 Missing Address Verification [LOW]

Description:

Certain functions lack a safety check in the address, the address-type arguments should include a zero-address test, otherwise, the contract's functionality may become inaccessible. In the constructor, the _outAsset, _vault, _indexSwapLibrary, _adapter, _accessController, _tokenMetadata and the _treasury arguments should be verified to be different from the address(0).

Code:

Listing 6: IndexSwap.sol

```
88  constructor(
89      string memory _name,
90      string memory _symbol,
91      address _outAsset,
92      address _vault,
93      uint256 _maxInvestmentAmount,
94      address _indexSwapLibrary,
95      address _adapter,
96      address _accessController,
97      address _tokenMetadata,
98      uint256 _feePointBasis,
99      address _treasury
100 ) TokenBase(_name, _symbol) {
101     vault = _vault;
102     outAsset = _outAsset; //As now we are tacking busd
103     MAX_INVESTMENTAMOUNT = _maxInvestmentAmount;
104     indexSwapLibrary = IIndexSwapLibrary(_indexSwapLibrary);
105     adapter = IAdapter(_adapter);
106     accessController = IAccessController(_accessController);
107     tokenMetadata = TokenMetadata(_tokenMetadata);
108     paused = false;
```

```
110         feePointBasis = _feePointBasis;
111         treasury = payable(_treasury);
112     }
```

Risk Level:

Likelihood – 1
Impact - 3

Recommendation:

We recommend that you make sure the addresses provided in the arguments are different from the address(0).

Status - Fixed

The Velvet team has fixed the issue by requiring the addresses provided in the arguments to be different from the address(0)

A.6 Owner Can Renounce Ownership [LOW]

Description:

Typically, the account that deploys the contract is also its owner. Consequently, the owner is able to engage in certain privileged activities in his own name. In smart contracts, the renounceOwnership function is used to renounce ownership, which means that if the contract's ownership has never been transferred, it will never have an Owner, rendering some owner-exclusive functionality unavailable.

Code:

Listing 7: IndexSwap.sol

```
32  contract TokenBase is ERC20Burnable, Ownable, ReentrancyGuard {
```

Risk Level:

Likelihood – 1
Impact - 3

Recommendation:

We recommend that you prevent the owner from calling renounceOwnership without first transferring ownership to a different address. Additionally, if you decide to use a multi-signature wallet, then the execution of the renounceOwnership will require for at least two or more users to be confirmed. Alternatively, you can disable Renounce Ownership functionality by overriding it.

Status - Acknowledged

The Velvet team has acknowledged the risk, stating that it is a risk with low-probability of occurrence.

A.7 Floating Pragma [INFORMATIONAL]

Description:

The contract makes use of the floating-point pragma 0.8.6. Contracts should be deployed using the same compiler version. Locking the pragma helps ensure that contracts will not unintentionally be deployed using another pragma, which in some cases may be an obsolete version, that may introduce issues to the contract system.

Code:

Listing 8: IndexSwap.sol

```
12 pragma solidity ^0.8.6;
```

Risk Level:

Likelihood – 1
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 - Fixed

The Velvet team has fixed the issue by locking the pragma version to 0.8.16.

B IndexFactory.sol

B.1 Missing Address Verification [LOW]

Description:

Certain functions lack a safety check in the address, the address-type arguments should include a zero-address test, otherwise, the contract's functionality may become inaccessible. In the constructor, the _uniswapRouter, _treasury, _indexSwapLibrary, _tokenMetadata, _baseAdapterAddress and the _baseRebalancingAddres arguments should be verified to be different from the address(0).

Code:

Listing 9: IndexSwapLibrary.sol

```
43  constructor(
44      address _uniswapRouter,
45      address _outAsset,
46      address _treasury,
47      address _indexSwapLibrary,
48      address _tokenMetadata,
49      address _baseAdapterAddress,
50      address _baseRebalancingAddres
51  ) {
52      require(_outAsset != address(0), "Invalid Out Asset");
```

```
53      uniswapRouter = _uniswapRouter;
54      outAsset = _outAsset;
55      treasury = _treasury;
56      indexSwapLibrary = _indexSwapLibrary;
57      tokenMetadata = _tokenMetadata;
58      noVtokenMetadata = Clones.clone(tokenMetadata);

60      baseRebalancingAddress = _baseRebalancingAddres;
61      baseAdapterAddress = _baseAdapterAddress;
62  }
```

Risk Level:

Likelihood – 1
Impact - 3

Recommendation:

We recommend that you make sure the addresses provided in the arguments are different from the address(0).

Status - Fixed

The Velvet team has fixed the issue by requiring the addresses provided in the arguments to be different from the address(0)

B.2 Owner Can Renounce Ownership [LOW]

Description:

Typically, the account that deploys the contract is also its owner. Consequently, the owner is able to engage in certain privileged activities in his own name. In smart contracts, the renounceOwnership function is used to renounce ownership, which means that if the contract's ownership has never been transferred, it will never have an Owner, rendering some owner-exclusive functionality unavailable.

Code:

Listing 10: IndexFactory.sol

```
12 contract IndexFactory is Ownable {
```

Risk Level:

Likelihood – 1
Impact - 3

Recommendation:

We recommend that you prevent the owner from calling renounceOwnership without first transferring ownership to a different address. Additionally, if you decide to use a multi-signature wallet, then the execution of the renounceOwnership will require for at least two or more users to be confirmed. Alternatively, you can disable Renounce Ownership functionality by overriding it.

Status - Acknowledged

The Velvet team has acknowledged the risk, stating that it is a risk with low-probability of occurrence.

B.3 Floating Pragma [INFORMATIONAL]

Description:

The contract makes use of the floating-point pragma 0.8.6. Contracts should be deployed using the same compiler version. Locking the pragma helps ensure that contracts will not unintentionally be deployed using another pragma, which in some cases may be an obsolete version, that may introduce issues to the contract system.

Code:

Listing 11: IndexFactory.sol

```
2 pragma solidity ^0.8.6;
```

Risk Level:

Likelihood – 1
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 - Fixed

The Velvet team has fixed the issue by locking the pragma version to 0.8.16.

C TokenMetadata.sol

C.1 Owner Can Renounce Ownership [LOW]

Description:

Typically, the account that deploys the contract is also its owner. Consequently, the owner is able to engage in certain privileged activities in his own name. In smart contracts, the renounceOwnership function is used to renounce ownership, which means that if the contract's ownership has never been transferred, it will never have an Owner, rendering some owner-exclusive functionality unavailable.

Code:

Listing 12: TokenMetadata.sol

```
16 contract TokenMetadata is Ownable {
```

Risk Level:

Likelihood – 1
Impact - 3

Recommendation:

We recommend that you prevent the owner from calling renounceOwnership without first transferring ownership to a different address. Additionally, if you decide to use a multi-signature wallet, then the execution of the renounceOwnership will require for at least two or more users to be confirmed. Alternatively, you can disable Renounce Ownership functionality by overriding it.

Status - Acknowledged

The Velvet team has acknowledged the risk, stating that it is a risk with low-probability of occurrence.

C.2 Floating Pragma [INFORMATIONAL]

Description:

The contract makes use of the floating-point pragma 0.8.6. Contracts should be deployed using the same compiler version. Locking the pragma helps ensure that contracts will not unintentionally be deployed using another pragma, which in some cases may be an obsolete version, that may introduce issues to the contract system.

Code:

Listing 13: TokenMetadata.sol

```
10 pragma solidity ^0.8.6;
```

Risk Level:

Likelihood – 1
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 - Fixed

The Velvet team has fixed the issue by locking the pragma version to 0.8.16.

D IndexSwapLibrary.sol

D.1 Missing Address Verification [LOW]

Description:

Certain functions lack a safety check in the address, the address-type arguments should include a zero-address test, otherwise, the contract's functionality may become inaccessible. In the constructor, the _oracle, _tokenMetadata and the _weth arguments should be verified to be different from the address(0).

Code:

Listing 14: IndexSwapLibrary.sol

```
29  constructor(
30      address _oracle,
31      address _weth,
32      TokenMetadata _tokenMetadata
33  ) {
34      oracle = IPriceOracle(_oracle);
35      wETH = _weth;
36      tokenMetadata = _tokenMetadata;
37  }
```

Risk Level:

Likelihood – 1
Impact - 3

Recommendation:

We recommend that you make sure the addresses provided in the arguments are different from the address(0).

Status - Fixed

The Velvet team has fixed the issue by requiring the addresses provided in the arguments to be different from the address(0)

D.2 Floating Pragma [INFORMATIONAL]

Description:

The contract makes use of the floating-point pragma 0.8.6. Contracts should be deployed using the same compiler version. Locking the pragma helps ensure that contracts will not unintentionally be deployed using another pragma, which in some cases may be an obsolete version, that may introduce issues to the contract system.

Code:

Listing 15: IndexSwapLibrary.sol

```
12 pragma solidity ^0.8.6;
```

Risk Level:

Likelihood – 1
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 - Fixed

The Velvet team has fixed the issue by locking the pragma version to 0.8.16.

E Rebalancing.sol

E.1 Missing Address Verification [LOW]

Description:

Certain functions lack a safety check in the address, the address-type arguments should include a zero-address test, otherwise, the contract's functionality may become inaccessible. In the init function, the _index, _indexSwapLibrary, _adapter, _accessController and the _tokenMetadata arguments should be verified to be different from the address(0).

Code:

Listing 16: Rebalancing.sol

```
58 function init(
59     IIndexSwap _index,
60     address _indexSwapLibrary,
61     address _adapter,
62     address _accessController,
63     address _tokenMetadata
64 ) external initializer {
65     index = IIndexSwap(_index);
66     indexSwapLibrary = IndexSwapLibrary(_indexSwapLibrary);
67     adapter = IAdapter(_adapter);
68     accessController = AccessController(_accessController);
69     tokenMetadata = TokenMetadata(_tokenMetadata);
70 }
```

Risk Level:

Likelihood – 1

Impact - 3

Recommendation:

We recommend that you make sure the addresses provided in the arguments are different from the address(0).

Status - Fixed

The Velvet team has fixed the issue by requiring the addresses provided in the arguments to be different from the address(0)

E.2 Floating Pragma [INFORMATIONAL]

Description:

The contract makes use of the floating-point pragma 0.8.6. Contracts should be deployed using the same compiler version. Locking the pragma helps ensure that contracts will not unintentionally be deployed using another pragma, which in some cases may be an obsolete version, that may introduce issues to the contract system.

Code:

Listing 17: Rebalancing.sol

```
15 pragma solidity ^0.8.6;
```

Risk Level:

Likelihood – 1
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 - Fixed

The Velvet team has fixed the issue by locking the pragma version to 0.8.16.

F PriceOracle.sol

F.1 Owner Can Renounce Ownership [LOW]

Description:

Typically, the account that deploys the contract is also its owner. Consequently, the owner is able to engage in certain privileged activities in his own name. In smart contracts, the renounceOwnership function is used to renounce ownership, which means that if the contract's ownership has never been transferred, it will never have an Owner, rendering some owner-exclusive functionality unavailable.

Code:

Listing 18: PriceOracle.sol

```
10 contract PriceOracle is Ownable {
```

Risk Level:

Likelihood – 1
Impact - 3

Recommendation:

We recommend that you prevent the owner from calling renounceOwnership without first transferring ownership to a different address. Additionally, if you decide to use a multi-signature wallet, then the execution of the renounceOwnership will require for at least two or more users to be confirmed. Alternatively, you can disable Renounce Ownership functionality by overriding it.

Status - Acknowledged

The Velvet team has acknowledged the risk, stating that it is a risk with low-probability of occurrence.

F.2 Floating Pragma [INFORMATIONAL]

Description:

The contract makes use of the floating-point pragma 0.8.6. Contracts should be deployed using the same compiler version. Locking the pragma helps ensure that contracts will not unintentionally be deployed using another pragma, which in some cases may be an obsolete version, that may introduce issues to the contract system.

Code:

Listing 19: PriceOracle.sol

```
2 pragma solidity ^0.8.6;
```

Risk Level:

Likelihood – 1
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 - Fixed

The Velvet team has fixed the issue by locking the pragma version to 0.8.16.

G VelvetSafeModule.sol

G.1 Missing Address Verification [LOW]

Description:

Certain functions lack a safety check in the address, the address-type arguments should include a zero-address test, otherwise, the contract's functionality may become inaccessible. In the constructor function, the _owner argument should be verified to be different from the address(0).

Code:

Listing 20: VelvetSafeModule.sol

```
22 constructor(address _owner) {
23     bytes memory initializeParams = abi.encode(_owner);
24     setUp(initializeParams);
25     moduleOwner = msg.sender;
26 }
```

Risk Level:

Likelihood – 1
Impact - 3

Recommendation:

We recommend that you make sure the addresses provided in the arguments are different from the address(0).

Status - Fixed

The Velvet team has fixed the issue by requiring the addresses provided in the arguments to be different from the address(0)

G.2 Floating Pragma [INFORMATIONAL]

Description:

The contract makes use of the floating-point pragma 0.8.6. Contracts should be deployed using the same compiler version. Locking the pragma helps ensure that contracts will not unintentionally be deployed using another pragma, which in some cases may be an obsolete version, that may introduce issues to the contract system.

Code:

Listing 21: VelvetSafeModule.sol

```
11 pragma solidity ^0.8.6;
```

Risk Level:

Likelihood – 1
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 - Fixed

The Velvet team has fixed the issue by locking the pragma version to 0.8.16.

H AccessController.sol

H.1 Floating Pragma [INFORMATIONAL]

Description:

The contract makes use of the floating-point pragma 0.8.6. Contracts should be deployed using the same compiler version. Locking the pragma helps ensure that contracts will not unintentionally be deployed using another pragma, which in some cases may be an obsolete version, that may introduce issues to the contract system.

Code:

Listing 22: PriceOracle.sol

```
2 pragma solidity ^0.8.6;
```

Risk Level:

Likelihood – 1
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 - Fixed

The Velvet team has fixed the issue by locking the pragma version to 0.8.16.

I Adapter.sol

I.1 Missing Address Verification [LOW]

Description:

Certain functions lack a safety check in the address, the address-type arguments should include a zero-address test, otherwise, the contract's functionality may become inaccessible. In the init function, the _accessController, _pancakeSwapAddress, _safe and the _tokenMetadata arguments should be verified to be different from the address(0).

Code:

Listing 23: Adapter.sol

```
46 function init(
47     address _accessController,
48     address _pancakeSwapAddress,
49     address _safe,
50     address _tokenMetadata
51 ) external initializer {
52     pancakeSwapRouter = IUniswapV2Router02(_pancakeSwapAddress);
53     accessController = AccessController(_accessController);
54     safe = IVault(_safe);
55     tokenMetadata = TokenMetadata(_tokenMetadata);
56 }
```

Risk Level:

Likelihood – 1
Impact - 3

Recommendation:

We recommend that you make sure the addresses provided in the arguments are different from the address(0).

Status - Fixed

The Velvet team has fixed the issue by requiring the addresses provided in the arguments to be different from the address(0)

I.2 Floating Pragma [INFORMATIONAL]

Description:

The contract makes use of the floating-point pragma 0.8.6. Contracts should be deployed using the same compiler version. Locking the pragma helps ensure that contracts will not unintentionally be deployed using another pragma, which in some cases may be an obsolete version, that may introduce issues to the contract system.

Code:

Listing 24: Adapter.sol

```
15 pragma solidity ^0.8.6;
```

Risk Level:

Likelihood – 1
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 - Fixed

The Velvet team has fixed the issue by locking the pragma version to 0.8.16.

J Vault.sol

J.1 Floating Pragma [INFORMATIONAL]

Description:

The contract makes use of the floating-point pragma 0.8.6. Contracts should be deployed using the same compiler version. Locking the pragma helps ensure that contracts will not unintentionally be deployed using another pragma, which in some cases may be an obsolete version, that may introduce issues to the contract system.

Code:

Listing 25: PriceOracle.sol

```
2 pragma solidity ^0.8.6;
```

Risk Level:

Likelihood – 1
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 - Fixed

The Velvet team has fixed the issue by locking the pragma version to 0.8.16.

4 Best Practices

BP.1 Use clone Instead Of new

Description:

There is one potential problem about using new to deploy contract, which is high gas costs. And that is where we can use cloning. When using clone, each contract will have its own state and simply uses the implementation contract as library. If you always deploy the same kind of contract, it is unnecessarily to waste gas costs for the bytecode. Any contract will have almost identical bytecode, so we do not have to store all bytecode again and again for each deployment.

Code:

Listing 26: IndexFactory.sol

```
72 AccessController accessController = new AccessController();
```

Listing 27: IndexFactory.sol

```
82 Vault _safe = new Vault();
```

Listing 28: IndexFactory.sol

```
92 IndexSwap indexSwap = new IndexSwap(
93     _name,
94     _symbol,
95     outAsset,
96     address(_safe),
97     maxInvestmentAmount,
98     indexSwapLibrary,
99     address(_adapter),
100    address(accessController),
101    tokenMetaDataInit,
102    feePointBasis,
103    treasury
```

5 Tests

Results:

```
No need to generate any newer typings.
56 chainId


  Tests for IndexFactory
    Tests for IndexFactory contract
0xC9E96fe9a02aAdacd48a7C9C5463Fb9db092dEB6 indexSwapAddress
indexSwap deployed to: 0xC9E96fe9a02aAdacd48a7C9C5463Fb9db092dEB6
      IndexFactory Contract
        should check Index token name and symbol
        initialize should revert if total Weights not equal 10,000 (40ms
          ↪ )
0xe9e7cea3dedca5984780bafc599bd69add087d56 busdInstance.address
0x2170Ed0880ac9A755fd29B2688956BD959F933F8 ethInstance.address
        Initialize IndexFund Tokens
BigNumber { value: "99632682673576004" }
        Invest 0.1BNB into Top10 fund (17001ms)
BigNumber { value: "2092273025076068549" }
        Invest 2BNB into Top10 fund (477ms)
BigNumber { value: "30885834812376837 21" }
        Invest 1BNB into Top10 fund (472ms)
        Investment should fail when contract is paused
        update Weights should revert if total Weights not equal 10,000
          ↪ (39ms)
        should revert to charge fees
        should Update Weights and Rebalance (1082ms)
        should Update Weights and Rebalance (1170ms)
        should Update Weights and Rebalance (470ms)
        should charge fees and treasury balance should increase (660ms)
        updateTokens should revert if total Weights not equal 10,000
```

```
        Non Rebalancing access address calling update function
        should update tokens (8080ms)
        withdrawal should revert when contract is paused
        should unpause
        should pause
        should revert unpause
        should unpause
        when withdraw fund more then balance
        should withdraw fund and burn index token successfully (1020ms)
        should withdraw tokens directly instead of BNB (2127ms)

  Tests for IndexSwap
    Tests for IndexSwap contract
Vault deployed to: 0xb12F0Ea584AcB5b676054e0F4f36EAa10d93A560
indexSwap deployed to: 0x0e03350D11F24839373262143B46D8d68E5E6649
     IndexSwap Contract
        should check Index token name and symbol
        initialize should revert if total Weights not equal 10,000
        Initialize IndexFund Tokens
BigNumber { value: "99632257869298897" }
        Invest 0.1BNB into Top10 fund (2095ms)
BigNumber { value: "2092264081331241238" }
        Invest 2BNB into Top10 fund (447ms)
BigNumber { value: "3088570272941503209" }
        Invest 1BNB into Top10 fund (749ms)
        Investment should fail when contract is paused
        update Weights should revert if total Weights not equal 10,000
        should revert to charge fees
        should Update Weights and Rebalance (762ms)
        should Update Weights and Rebalance (852ms)
        should Update Weights and Rebalance (475ms)
        should charge fees and treasury balance should increase (279ms)
        updateTokens should revert if total Weights not equal 10,000
        owner should be able to add asset manager
```

```
        non owner should not be able to add asset manager
        new asset manager should update tokens (1918ms)
        withdrawal should revert when contract is paused
        should unpause
        should pause
        should revert unpause
        should unpause
        when withdraw fund more then balance
        should withdraw fund and burn index token successfully (982ms)
        should withdraw tokens directly instead of BNB (704ms)

  Tests for priceOracle
    Tests for priceOracle contract
      priceOracle Contract
BigNumber { value: "57203060343969060000" }
        Get ETH/WBNB price (1380ms)
BigNumber { value: "12704811147575256000" }
        Get BTC/ETH price (1878ms)
BigNumber { value: "3599928816839989" }
        Get BUSD/WBNB price (1711ms)
BigNumber { value: "2008720000000" }
        Get BTC/USD price (1237ms)
BigNumber { value: "20087200000000000000000" }
        Get BTC/USD price
BigNumber { value: "1592807622410000000000" }
        Get ETH/USD price
BigNumber { value: "1000000000000000000" }
        Get BUSD/USD price
BigNumber { value: "9996408000000000000" }
        Get DAI/USD price
BigNumber { value: "2788906386300000000000" }
        Get WBNB/USD price
BigNumber { value: "61590000000000000" }
        Get DOGE/USD price (1690ms)
```

```
BigNumber { value: "3585634874344724" }
        Get USD/WBNB price
BigNumber { value: "3585634874344724" }
        Get USD/WBNB price


  61 passing (1m)
```

6 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:

```
'npx hardhat compile --force' running
Generating typings for: 60 artifacts in dir: typechain for target:
   ↪ ethers-v5
Successfully generated 104 typings!
Compiled 58 Solidity files successfully

Solidity 0.8.10 is not fully supported yet. You can still use Hardhat,
   ↪ but some features, like stack traces, might not work correctly.

Learn more at https://hardhat.org/hardhat-runner/docs/reference/solidity
   ↪ -support



Adapter._swapETHToToken(address,uint256,address,uint256) (contracts/core
   ↪ /Adapter.sol#108-148) sends eth to arbitrary user
 Dangerous calls:
 - IWETH(t).deposit{value: swapAmount}() (contracts/core/Adapter.sol
    ↪ #119)
 - swapResult = pancakeSwapRouter.swapExactETHForTokens{value:
    ↪ swapAmount}(getSlippage(swapAmount,_slippage,getPathForETH(t)),
```

```
    ↪ getPathForETH(t),address(this),block.timestamp)[1] (contracts/
    ↪ core/Adapter.sol#128-135)
 - swapResult = pancakeSwapRouter.swapExactETHForTokens{value:
    ↪ swapAmount}(getSlippage(swapAmount,_slippage,getPathForETH(t)),
    ↪ getPathForETH(t),to,block.timestamp)[1] (contracts/core/Adapter.
    ↪ sol#138-145)
Adapter._swapTokenToETH(address,uint256,address,uint256) (contracts/core
   ↪ /Adapter.sol#158-216) sends eth to arbitrary user
 Dangerous calls:
 - (success) = address(to).call{value: swapResult}() (contracts/core/
    ↪ Adapter.sol#169)
 - (success) = address(to).call{value: swapAmount}() (contracts/core/
    ↪ Adapter.sol#203)
Adapter.lendBNB(address,address,uint256,address) (contracts/core/Adapter
   ↪ .sol#238-255) sends eth to arbitrary user
 Dangerous calls:
 - vToken.mint{value: _amount}() (contracts/core/Adapter.sol#252)
Adapter.redeemBNB(address,uint256,address) (contracts/core/Adapter.sol
   ↪ #277-296) sends eth to arbitrary user
 Dangerous calls:
 - (success) = address(_to).call{value: address(this).balance}() (
    ↪ contracts/core/Adapter.sol#291-293)
Rebalancing.buyTokens(uint256[],uint256[],uint256,uint256) (contracts/
   ↪ rebalance/Rebalancing.sol#179-203) sends eth to arbitrary user
 Dangerous calls:
 - adapter._swapETHToToken{value: swapAmount}(index.getTokens()[i],
    ↪ swapAmount,index.vault(),_slippage) (contracts/rebalance/
    ↪ Rebalancing.sol#195-200)
Rebalancing.feeModule() (contracts/rebalance/Rebalancing.sol#365-440)
   ↪ sends eth to arbitrary user
 Dangerous calls:
 - (success) = address(index.treasury()).call{value: amount}() (
    ↪ contracts/rebalance/Rebalancing.sol#406-408)
```

```
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #functions-that-send-ether-to-arbitrary-destinations

Initializable is re-used:
 - Initializable (node_modules/@openzeppelin/contracts-upgradeable/proxy
     ↪ /utils/Initializable.sol#57-149)
 - Initializable (node_modules/@openzeppelin/contracts/proxy/utils/
     ↪ Initializable.sol#57-149)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #name-reused

Reentrancy in IndexSwap.investInFund(uint256) (contracts/core/IndexSwap.
    ↪ sol#170-226):
 External calls:
 - (tokenBalanceInBNB,vaultBalance) = indexSwapLibrary.
     ↪ getTokenAndVaultBalance(IIndexSwap(address(this))) (contracts/
     ↪ core/IndexSwap.sol#184-185)
 - investedAmountAfterSlippage = _swapETHToTokens(tokenAmount,amount,
     ↪ _slippage) (contracts/core/IndexSwap.sol#194-198)
  - swapResult = adapter._swapETHToToken{value: swapAmount}(t,swapAmount
      ↪ ,vault,_slippage) (contracts/core/IndexSwap.sol#349-354)
 External calls sending eth:
 - investedAmountAfterSlippage = _swapETHToTokens(tokenAmount,amount,
     ↪ _slippage) (contracts/core/IndexSwap.sol#194-198)
  - swapResult = adapter._swapETHToToken{value: swapAmount}(t,swapAmount
      ↪ ,vault,_slippage) (contracts/core/IndexSwap.sol#349-354)
 State variables written after the call(s):
 - _mint(msg.sender,tokenAmount) (contracts/core/IndexSwap.sol#217)
  - _totalSupply += amount (node_modules/@openzeppelin/contracts/token/
      ↪ ERC20/ERC20.sol#262)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #reentrancy-vulnerabilities
```

```
OwnableUpgradeable.__gap (node_modules/@openzeppelin/contracts-
    ↪ upgradeable/access/OwnableUpgradeable.sol#87) shadows:
 - ContextUpgradeable.__gap (node_modules/@openzeppelin/contracts-
     ↪ upgradeable/utils/ContextUpgradeable.sol#36)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #state-variable-shadowing

Adapter._swapETHToToken(address,uint256,address,uint256) (contracts/core
    ↪ /Adapter.sol#108-148) ignores return value by IWETH(t).transfer(
    ↪ to,swapAmount) (contracts/core/Adapter.sol#123)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #unchecked-transfer

InputData (contracts/vault/VelvetSafeModule.sol#15-17) has incorrect
    ↪ ERC20 function interface:InputData.transfer(address,uint256) (
    ↪ contracts/vault/VelvetSafeModule.sol#16)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #incorrect-erc20-interface

Adapter.lendToken(address,address,uint256,address) (contracts/core/
    ↪ Adapter.sol#219-236) uses a dangerous strict equality:
 - assert(bool)(vToken.mint(_amount) == 0) (contracts/core/Adapter.sol
     ↪ #233)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #dangerous-strict-equalities

Contract locking ether found:
 Contract ERC20Mock (contracts/mock/ERC20Mock.sol#6-39) has payable
     ↪ functions:
  - ERC20Mock.constructor(string,string,address,uint256) (contracts/mock
      ↪ /ERC20Mock.sol#7-14)
 But does not have a function to withdraw the ether
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #contracts-that-lock-ether
```

```
IndexFactory.validIndexId(uint256) (contracts/IndexFactory.sol#168-172)
    ↪ contains a tautology or contradiction:
 - indexfundId >= 0 && indexfundId <= IndexSwapInfolList.length - 1 (
     ↪ contracts/IndexFactory.sol#169)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #tautology-or-contradiction

Adapter._swapTokenToETH(address,uint256,address,uint256).success_scope_0
    ↪  (contracts/core/Adapter.sol#203) is a local variable never
    ↪ initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #uninitialized-local-variables

Rebalancing.sellTokens(uint256[],uint256[],uint256) (contracts/rebalance
    ↪ /Rebalancing.sol#114-172) ignores return value by adapter.
    ↪ redeemBNB(tokenMetadata.vTokens(index.getTokens()[i]),swapAmount,
    ↪ address(this)) (contracts/rebalance/Rebalancing.sol#145-149)
Rebalancing.sellTokens(uint256[],uint256[],uint256) (contracts/rebalance
    ↪ /Rebalancing.sol#114-172) ignores return value by adapter.
    ↪ _swapTokenToETH(index.getTokens()[i],swapAmount,address(this),
    ↪ _slippage) (contracts/rebalance/Rebalancing.sol#160-165)
Rebalancing.buyTokens(uint256[],uint256[],uint256,uint256) (contracts/
    ↪ rebalance/Rebalancing.sol#179-203) ignores return value by
    ↪ adapter._swapETHToToken{value: swapAmount}(index.getTokens()[i],
    ↪ swapAmount,index.vault(),_slippage) (contracts/rebalance/
    ↪ Rebalancing.sol#195-200)
Rebalancing.updateTokens(address[],uint96[],uint256) (contracts/
    ↪ rebalance/Rebalancing.sol#291-362) ignores return value by
    ↪ adapter.redeemBNB(tokenMetadata.vTokens(index.getTokens()[
    ↪ i_scope_0]),tokenBalance,address(this)) (contracts/rebalance/
    ↪ Rebalancing.sol#328-332)
Rebalancing.updateTokens(address[],uint96[],uint256) (contracts/
    ↪ rebalance/Rebalancing.sol#291-362) ignores return value by
```

```
    ↪ adapter._swapTokenToETH(index.getTokens()[i_scope_0],tokenBalance
    ↪ ,address(this),_slippage) (contracts/rebalance/Rebalancing.sol
    ↪ #343-348)
Rebalancing.feeModule() (contracts/rebalance/Rebalancing.sol#365-440)
    ↪ ignores return value by adapter.redeemBNB(tokenMetadata.vTokens(
    ↪ index.getTokens()[i]),amount,index.treasury()) (contracts/
    ↪ rebalance/Rebalancing.sol#391-395)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #unused-return

TokenBase.constructor(string,string)._name (contracts/core/IndexSwap.sol
    ↪ #33) shadows:
 - ERC20._name (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.
     ↪ sol#42) (state variable)
TokenBase.constructor(string,string)._symbol (contracts/core/IndexSwap.
    ↪ sol#33) shadows:
 - ERC20._symbol (node_modules/@openzeppelin/contracts/token/ERC20/ERC20
     ↪ .sol#43) (state variable)
IndexSwap.constructor(string,string,address,address,uint256,address,
    ↪ address,address,address,uint256,address)._name (contracts/core/
    ↪ IndexSwap.sol#89) shadows:
 - ERC20._name (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.
     ↪ sol#42) (state variable)
IndexSwap.constructor(string,string,address,address,uint256,address,
    ↪ address,address,address,uint256,address)._symbol (contracts/core/
    ↪ IndexSwap.sol#90) shadows:
 - ERC20._symbol (node_modules/@openzeppelin/contracts/token/ERC20/ERC20
     ↪ .sol#43) (state variable)
ERC20Mock.constructor(string,string,address,uint256).name (contracts/
    ↪ mock/ERC20Mock.sol#8) shadows:
 - ERC20.name() (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.
     ↪ sol#62-64) (function)
 - IERC20Metadata.name() (node_modules/@openzeppelin/contracts/token/
     ↪ ERC20/extensions/IERC20Metadata.sol#17) (function)
```

```
ERC20Mock.constructor(string,string,address,uint256).symbol (contracts/
    ↪ mock/ERC20Mock.sol#9) shadows:
 - ERC20.symbol() (node_modules/@openzeppelin/contracts/token/ERC20/
     ↪ ERC20.sol#70-72) (function)
 - IERC20Metadata.symbol() (node_modules/@openzeppelin/contracts/token/
     ↪ ERC20/extensions/IERC20Metadata.sol#22) (function)
VelvetSafeModule.constructor(address)._owner (contracts/vault/
    ↪ VelvetSafeModule.sol#22) shadows:
 - OwnableUpgradeable._owner (node_modules/@openzeppelin/contracts-
     ↪ upgradeable/access/OwnableUpgradeable.sol#22) (state variable)
VelvetSafeModule.setUp(bytes)._owner (contracts/vault/VelvetSafeModule.
    ↪ sol#45) shadows:
 - OwnableUpgradeable._owner (node_modules/@openzeppelin/contracts-
     ↪ upgradeable/access/OwnableUpgradeable.sol#22) (state variable)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #local-variable-shadowing

Vault.transferModuleOwnership(address) (contracts/vault/Vault.sol#19-25)
    ↪  should emit an event for:
 - moduleOwner = newOwner (contracts/vault/Vault.sol#24)
VelvetSafeModule.transferModuleOwnership(address) (contracts/vault/
    ↪ VelvetSafeModule.sol#33-39) should emit an event for:
 - moduleOwner = newOwner (contracts/vault/VelvetSafeModule.sol#38)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #missing-events-access-control

IndexFactory.constructor(address,address,address,address,address,address
    ↪ ,address)._uniswapRouter (contracts/IndexFactory.sol#44) lacks a
    ↪ zero-check on :
  - uniswapRouter = _uniswapRouter (contracts/IndexFactory.sol#53)
IndexFactory.constructor(address,address,address,address,address,address
    ↪ ,address)._treasury (contracts/IndexFactory.sol#46) lacks a zero-
    ↪ check on :
  - treasury = _treasury (contracts/IndexFactory.sol#55)
```

```
IndexFactory.constructor(address,address,address,address,address,address
    ↪ ,address)._indexSwapLibrary (contracts/IndexFactory.sol#47) lacks
    ↪  a zero-check on :
  - indexSwapLibrary = _indexSwapLibrary (contracts/IndexFactory.sol#56)
IndexFactory.constructor(address,address,address,address,address,address
    ↪ ,address)._tokenMetadata (contracts/IndexFactory.sol#48) lacks a
    ↪ zero-check on :
  - tokenMetadata = _tokenMetadata (contracts/IndexFactory.sol#57)
IndexFactory.constructor(address,address,address,address,address,address
    ↪ ,address)._baseRebalancingAddres (contracts/IndexFactory.sol#50)
    ↪ lacks a zero-check on :
  - baseRebalancingAddress = _baseRebalancingAddres (contracts/
      ↪ IndexFactory.sol#60)
IndexFactory.constructor(address,address,address,address,address,address
    ↪ ,address)._baseAdapterAddress (contracts/IndexFactory.sol#49)
    ↪ lacks a zero-check on :
  - baseAdapterAddress = _baseAdapterAddress (contracts/IndexFactory.sol
      ↪ #61)
IndexFactory.setOutAsset(address)._outAsset (contracts/IndexFactory.sol
    ↪ #195) lacks a zero-check on :
  - outAsset = _outAsset (contracts/IndexFactory.sol#197)
Adapter._swapTokenToETH(address,uint256,address,uint256).to (contracts/
    ↪ core/Adapter.sol#161) lacks a zero-check on :
  - (success) = address(to).call{value: swapResult}() (contracts/core/
      ↪ Adapter.sol#169)
  - (success) = address(to).call{value: swapAmount}() (contracts/core/
      ↪ Adapter.sol#203)
IndexSwap.constructor(string,string,address,address,uint256,address,
    ↪ address,address,address,uint256,address)._vault (contracts/core/
    ↪ IndexSwap.sol#92) lacks a zero-check on :
  - vault = _vault (contracts/core/IndexSwap.sol#101)
IndexSwap.constructor(string,string,address,address,uint256,address,
    ↪ address,address,address,uint256,address)._outAsset (contracts/
    ↪ core/IndexSwap.sol#91) lacks a zero-check on :
```

```
  - outAsset = _outAsset (contracts/core/IndexSwap.sol#102)
IndexSwap.constructor(string,string,address,address,uint256,address,
    ↪ address,address,address,uint256,address)._treasury (contracts/
    ↪ core/IndexSwap.sol#99) lacks a zero-check on :
  - treasury = address(_treasury) (contracts/core/IndexSwap.sol#111)
IndexSwap.updateTreasury(address)._newTreasury (contracts/core/IndexSwap
    ↪ .sol#420) lacks a zero-check on :
  - treasury = _newTreasury (contracts/core/IndexSwap.sol#424)
IndexSwapLibrary.constructor(address,address,TokenMetadata)._weth (
    ↪ contracts/core/IndexSwapLibrary.sol#31) lacks a zero-check on :
  - wETH = _weth (contracts/core/IndexSwapLibrary.sol#35)
Vault.executeTransactionETH(address,uint256)._to (contracts/vault/Vault.
    ↪ sol#27) lacks a zero-check on :
  - (successReturn) = address(_to).call{value: value}() (contracts/vault
      ↪ /Vault.sol#32)
Module.setAvatar(address)._avatar (node_modules/@gnosis.pm/zodiac/
    ↪ contracts/core/Module.sol#23) lacks a zero-check on :
  - avatar = _avatar (node_modules/@gnosis.pm/zodiac/contracts/core/
      ↪ Module.sol#25)
Module.setTarget(address)._target (node_modules/@gnosis.pm/zodiac/
    ↪ contracts/core/Module.sol#31) lacks a zero-check on :
  - target = _target (node_modules/@gnosis.pm/zodiac/contracts/core/
      ↪ Module.sol#33)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #missing-zero-address-validation

IndexSwap.withdrawFund(uint256,uint256,bool) (contracts/core/IndexSwap.
    ↪ sol#233-324) has external calls inside a loop: tokenBalance =
    ↪ indexSwapLibrary.getTokenBalance(IIndexSwap(address(this)),
    ↪ _tokens[i],adapter.getETH() == _tokens[i]) (contracts/core/
    ↪ IndexSwap.sol#250-254)
IndexSwap.withdrawFund(uint256,uint256,bool) (contracts/core/IndexSwap.
    ↪ sol#233-324) has external calls inside a loop: _tokens[i] ==
    ↪ adapter.getETH() (contracts/core/IndexSwap.sol#261)
```

```
IndexSwap.withdrawFund(uint256,uint256,bool) (contracts/core/IndexSwap.
    ↪ sol#233-324) has external calls inside a loop: tokenMetadata.
    ↪ vTokens(_tokens[i]) != address(0) (contracts/core/IndexSwap.sol
    ↪ #262)
IndexSwap.withdrawFund(uint256,uint256,bool) (contracts/core/IndexSwap.
    ↪ sol#233-324) has external calls inside a loop: adapter.
    ↪ _pullFromVault(IIndexSwap(address(this)),_tokens[i],amount,
    ↪ address(adapter)) (contracts/core/IndexSwap.sol#263-268)
IndexSwap.withdrawFund(uint256,uint256,bool) (contracts/core/IndexSwap.
    ↪ sol#233-324) has external calls inside a loop: bal = adapter.
    ↪ redeemBNB(tokenMetadata.vTokens(_tokens[i]),amount,msg.sender) (
    ↪ contracts/core/IndexSwap.sol#270-274)
IndexSwap.withdrawFund(uint256,uint256,bool) (contracts/core/IndexSwap.
    ↪ sol#233-324) has external calls inside a loop: adapter.
    ↪ _pullFromVault(IIndexSwap(address(this)),_tokens[i],amount,
    ↪ address(this)) (contracts/core/IndexSwap.sol#278-283)
IndexSwap.withdrawFund(uint256,uint256,bool) (contracts/core/IndexSwap.
    ↪ sol#233-324) has external calls inside a loop: IWETH(_tokens[i]).
    ↪ withdraw(amount) (contracts/core/IndexSwap.sol#285)
IndexSwap.withdrawFund(uint256,uint256,bool) (contracts/core/IndexSwap.
    ↪ sol#233-324) has external calls inside a loop: (success) =
    ↪ address(msg.sender).call{value: amount}() (contracts/core/
    ↪ IndexSwap.sol#288-290)
IndexSwap.withdrawFund(uint256,uint256,bool) (contracts/core/IndexSwap.
    ↪ sol#233-324) has external calls inside a loop: adapter.
    ↪ _pullFromVault(IIndexSwap(address(this)),_tokens[i],amount,
    ↪ address(adapter)) (contracts/core/IndexSwap.sol#294-299)
IndexSwap.withdrawFund(uint256,uint256,bool) (contracts/core/IndexSwap.
    ↪ sol#233-324) has external calls inside a loop: sw = adapter.
    ↪ _swapTokenToETH(_tokens[i],amount,msg.sender,_slippage) (
    ↪ contracts/core/IndexSwap.sol#300-305)
IndexSwap.withdrawFund(uint256,uint256,bool) (contracts/core/IndexSwap.
    ↪ sol#233-324) has external calls inside a loop: adapter.
    ↪ _pullFromVault(IIndexSwap(address(this)),_tokens[i],amount,msg.
```

```
    ↪ sender) (contracts/core/IndexSwap.sol#310-315)
IndexSwapLibrary.getTokenAndVaultBalance(IIndexSwap) (contracts/core/
    ↪ IndexSwapLibrary.sol#45-105) has external calls inside a loop: i
    ↪ < _index.getTokens().length (contracts/core/IndexSwapLibrary.sol
    ↪ #55)
IndexSwapLibrary.getTokenAndVaultBalance(IIndexSwap) (contracts/core/
    ↪ IndexSwapLibrary.sol#45-105) has external calls inside a loop:
    ↪ tokenMetadata.vTokens(_index.getTokens()[i]) != address(0) (
    ↪ contracts/core/IndexSwapLibrary.sol#60)
IndexSwapLibrary.getTokenAndVaultBalance(IIndexSwap) (contracts/core/
    ↪ IndexSwapLibrary.sol#45-105) has external calls inside a loop:
    ↪ _index.getTokens()[i] != wETH (contracts/core/IndexSwapLibrary.
    ↪ sol#62)
IndexSwapLibrary.getTokenAndVaultBalance(IIndexSwap) (contracts/core/
    ↪ IndexSwapLibrary.sol#45-105) has external calls inside a loop:
    ↪ token = VBep20Interface(tokenMetadata.vTokens(_index.getTokens()[
    ↪ i])) (contracts/core/IndexSwapLibrary.sol#63-65)
IndexSwapLibrary.getTokenAndVaultBalance(IIndexSwap) (contracts/core/
    ↪ IndexSwapLibrary.sol#45-105) has external calls inside a loop:
    ↪ tokenBalance = token.balanceOfUnderlying(_index.vault()) (
    ↪ contracts/core/IndexSwapLibrary.sol#66-68)
IndexSwapLibrary.getTokenAndVaultBalance(IIndexSwap) (contracts/core/
    ↪ IndexSwapLibrary.sol#45-105) has external calls inside a loop:
    ↪ tokenBalanceUSD = _getTokenAmountInUSD(_index.getTokens()[i],
    ↪ tokenBalance) (contracts/core/IndexSwapLibrary.sol#70-73)
IndexSwapLibrary._getTokenAmountInUSD(address,uint256) (contracts/core/
    ↪ IndexSwapLibrary.sol#162-168) has external calls inside a loop:
    ↪ amountInUSD = oracle.getPriceTokenUSD(t,amount) (contracts/core/
    ↪ IndexSwapLibrary.sol#167)
IndexSwapLibrary.getTokenAndVaultBalance(IIndexSwap) (contracts/core/
    ↪ IndexSwapLibrary.sol#45-105) has external calls inside a loop:
    ↪ token_scope_0 = IVBNB(tokenMetadata.vTokens(_index.getTokens()[i
    ↪ ])) (contracts/core/IndexSwapLibrary.sol#75-77)
```

```
IndexSwapLibrary.getTokenAndVaultBalance(IIndexSwap) (contracts/core/
    ↪ IndexSwapLibrary.sol#45-105) has external calls inside a loop:
    ↪ tokenBalanceUnderlying = token_scope_0.balanceOfUnderlying(_index
    ↪ .vault()) (contracts/core/IndexSwapLibrary.sol#78-79)
IndexSwapLibrary.getTokenAndVaultBalance(IIndexSwap) (contracts/core/
    ↪ IndexSwapLibrary.sol#45-105) has external calls inside a loop:
    ↪ tokenBalanceUSD = _getTokenAmountInUSD(_index.getTokens()[i],
    ↪ tokenBalanceUnderlying) (contracts/core/IndexSwapLibrary.sol
    ↪ #81-84)
IndexSwapLibrary.getTokenAndVaultBalance(IIndexSwap) (contracts/core/
    ↪ IndexSwapLibrary.sol#45-105) has external calls inside a loop:
    ↪ tokenBalance = IERC20(_index.getTokens()[i]).balanceOf(_index.
    ↪ vault()) (contracts/core/IndexSwapLibrary.sol#87-89)
IndexSwapLibrary.getTokenAndVaultBalance(IIndexSwap) (contracts/core/
    ↪ IndexSwapLibrary.sol#45-105) has external calls inside a loop:
    ↪ tokenBalanceUSD = _getTokenAmountInUSD(_index.getTokens()[i],
    ↪ tokenBalance) (contracts/core/IndexSwapLibrary.sol#90-93)
IndexSwapLibrary.calculateSwapAmounts(IIndexSwap,uint256,uint256[],
    ↪ uint256) (contracts/core/IndexSwapLibrary.sol#139-154) has
    ↪ external calls inside a loop: i < _index.getTokens().length (
    ↪ contracts/core/IndexSwapLibrary.sol#147)
Rebalancing.updateTokens(address[],uint96[],uint256) (contracts/
    ↪ rebalance/Rebalancing.sol#291-362) has external calls inside a
    ↪ loop: i_scope_0 < index.getTokens().length (contracts/rebalance/
    ↪ Rebalancing.sol#308)
Rebalancing.updateTokens(address[],uint96[],uint256) (contracts/
    ↪ rebalance/Rebalancing.sol#291-362) has external calls inside a
    ↪ loop: tokenBalance = indexSwapLibrary.getTokenBalance(index,index
    ↪ .getTokens()[i_scope_0],adapter.getETH() == index.getTokens()[
    ↪ i_scope_0]) (contracts/rebalance/Rebalancing.sol#311-315)
Rebalancing.updateTokens(address[],uint96[],uint256) (contracts/
    ↪ rebalance/Rebalancing.sol#291-362) has external calls inside a
    ↪ loop: index.getTokens()[i_scope_0] == adapter.getETH() (contracts
    ↪ /rebalance/Rebalancing.sol#317)
```

```
Rebalancing.updateTokens(address[],uint96[],uint256) (contracts/
    ↪ rebalance/Rebalancing.sol#291-362) has external calls inside a
    ↪ loop: adapter._pullFromVault(index,index.getTokens()[i_scope_0],
    ↪ tokenBalance,address(this)) (contracts/rebalance/Rebalancing.sol
    ↪ #318-323)
Rebalancing.updateTokens(address[],uint96[],uint256) (contracts/
    ↪ rebalance/Rebalancing.sol#291-362) has external calls inside a
    ↪ loop: tokenMetadata.vTokens(index.getTokens()[i_scope_0]) !=
    ↪ address(0) (contracts/rebalance/Rebalancing.sol#325-326)
Rebalancing.updateTokens(address[],uint96[],uint256) (contracts/
    ↪ rebalance/Rebalancing.sol#291-362) has external calls inside a
    ↪ loop: adapter.redeemBNB(tokenMetadata.vTokens(index.getTokens()[
    ↪ i_scope_0]),tokenBalance,address(this)) (contracts/rebalance/
    ↪ Rebalancing.sol#328-332)
Rebalancing.updateTokens(address[],uint96[],uint256) (contracts/
    ↪ rebalance/Rebalancing.sol#291-362) has external calls inside a
    ↪ loop: index.deleteRecord(index.getTokens()[i_scope_0]) (contracts
    ↪ /rebalance/Rebalancing.sol#351)
Rebalancing.updateTokens(address[],uint96[],uint256) (contracts/
    ↪ rebalance/Rebalancing.sol#291-362) has external calls inside a
    ↪ loop: IWETH(index.getTokens()[i_scope_0]).withdraw(tokenBalance)
    ↪ (contracts/rebalance/Rebalancing.sol#334)
Rebalancing.updateTokens(address[],uint96[],uint256) (contracts/
    ↪ rebalance/Rebalancing.sol#291-362) has external calls inside a
    ↪ loop: adapter._pullFromVault(index,index.getTokens()[i_scope_0],
    ↪ tokenBalance,address(adapter)) (contracts/rebalance/Rebalancing.
    ↪ sol#337-342)
Rebalancing.updateTokens(address[],uint96[],uint256) (contracts/
    ↪ rebalance/Rebalancing.sol#291-362) has external calls inside a
    ↪ loop: adapter._swapTokenToETH(index.getTokens()[i_scope_0],
    ↪ tokenBalance,address(this),_slippage) (contracts/rebalance/
    ↪ Rebalancing.sol#343-348)
Rebalancing.feeModule() (contracts/rebalance/Rebalancing.sol#365-440)
    ↪ has external calls inside a loop: i < index.getTokens().length (
```

```
    ↪ contracts/rebalance/Rebalancing.sol#371)
Rebalancing.feeModule() (contracts/rebalance/Rebalancing.sol#365-440)
    ↪ has external calls inside a loop: tokenBalance = indexSwapLibrary
    ↪ .getTokenBalance(index,index.getTokens()[i],adapter.getETH() ==
    ↪ index.getTokens()[i]) (contracts/rebalance/Rebalancing.sol
    ↪ #372-376)
Rebalancing.feeModule() (contracts/rebalance/Rebalancing.sol#365-440)
    ↪ has external calls inside a loop: amount = tokenBalance.mul(index
    ↪ .feePointBasis()).div(10_000) (contracts/rebalance/Rebalancing.
    ↪ sol#378-380)
Rebalancing.feeModule() (contracts/rebalance/Rebalancing.sol#365-440)
    ↪ has external calls inside a loop: index.getTokens()[i] == adapter
    ↪ .getETH() (contracts/rebalance/Rebalancing.sol#382)
Rebalancing.feeModule() (contracts/rebalance/Rebalancing.sol#365-440)
    ↪ has external calls inside a loop: tokenMetadata.vTokens(index.
    ↪ getTokens()[i]) != address(0) (contracts/rebalance/Rebalancing.
    ↪ sol#383)
Rebalancing.feeModule() (contracts/rebalance/Rebalancing.sol#365-440)
    ↪ has external calls inside a loop: adapter._pullFromVault(index,
    ↪ index.getTokens()[i],amount,address(adapter)) (contracts/
    ↪ rebalance/Rebalancing.sol#384-389)
Rebalancing.feeModule() (contracts/rebalance/Rebalancing.sol#365-440)
    ↪ has external calls inside a loop: adapter.redeemBNB(tokenMetadata
    ↪ .vTokens(index.getTokens()[i]),amount,index.treasury()) (
    ↪ contracts/rebalance/Rebalancing.sol#391-395)
Rebalancing.feeModule() (contracts/rebalance/Rebalancing.sol#365-440)
    ↪ has external calls inside a loop: FeeCharged(block.timestamp,
    ↪ index.getTokens()[i],amount) (contracts/rebalance/Rebalancing.sol
    ↪ #436)
Rebalancing.feeModule() (contracts/rebalance/Rebalancing.sol#365-440)
    ↪ has external calls inside a loop: adapter._pullFromVault(index,
    ↪ index.getTokens()[i],amount,address(this)) (contracts/rebalance/
    ↪ Rebalancing.sol#397-402)
```

```
Rebalancing.feeModule() (contracts/rebalance/Rebalancing.sol#365-440)
    ↪ has external calls inside a loop: IWETH(index.getTokens()[i]).
    ↪ withdraw(amount) (contracts/rebalance/Rebalancing.sol#404)
Rebalancing.feeModule() (contracts/rebalance/Rebalancing.sol#365-440)
    ↪ has external calls inside a loop: (success) = address(index.
    ↪ treasury()).call{value: amount}() (contracts/rebalance/
    ↪ Rebalancing.sol#406-408)
Rebalancing.feeModule() (contracts/rebalance/Rebalancing.sol#365-440)
    ↪ has external calls inside a loop: tokenMetadata.vTokens(index.
    ↪ getTokens()[i]) != address(0) (contracts/rebalance/Rebalancing.
    ↪ sol#412)
Rebalancing.feeModule() (contracts/rebalance/Rebalancing.sol#365-440)
    ↪ has external calls inside a loop: adapter._pullFromVault(index,
    ↪ index.getTokens()[i],amount,address(adapter)) (contracts/
    ↪ rebalance/Rebalancing.sol#413-418)
Rebalancing.feeModule() (contracts/rebalance/Rebalancing.sol#365-440)
    ↪ has external calls inside a loop: adapter.redeemToken(
    ↪ tokenMetadata.vTokens(index.getTokens()[i]),index.getTokens()[i],
    ↪ amount,index.treasury()) (contracts/rebalance/Rebalancing.sol
    ↪ #420-425)
Rebalancing.feeModule() (contracts/rebalance/Rebalancing.sol#365-440)
    ↪ has external calls inside a loop: adapter._pullFromVault(index,
    ↪ index.getTokens()[i],amount,index.treasury()) (contracts/
    ↪ rebalance/Rebalancing.sol#427-432)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ /#calls-inside-a-loop

Variable 'Adapter._swapTokenToETH(address,uint256,address,uint256).
    ↪ success (contracts/core/Adapter.sol#169)' in Adapter.
    ↪ _swapTokenToETH(address,uint256,address,uint256) (contracts/core/
    ↪ Adapter.sol#158-216) potentially used before declaration: (
    ↪ success) = address(to).call{value: swapAmount}() (contracts/core/
    ↪ Adapter.sol#203)
```

```
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #pre-declaration-usage-of-local-variables

Reentrancy in IndexFactory.createIndex(string,string,uint256,uint256,
    ↪ bool) (contracts/IndexFactory.sol#64-166):
 External calls:
 - _adapter.init(address(accessController),uniswapRouter,address(_safe),
     ↪ tokenMetaDataInit) (contracts/IndexFactory.sol#85-90)
 - rebalancing.init(IIndexSwap(address(indexSwap)),indexSwapLibrary,
     ↪ address(_adapter),address(accessController),tokenMetaDataInit) (
     ↪ contracts/IndexFactory.sol#121-127)
 State variables written after the call(s):
 - IndexSwapInfolList.push(IndexSwaplInfo(address(indexSwap),address(
     ↪ rebalancing),owner())) (contracts/IndexFactory.sol#129-131)
Reentrancy in IndexSwap.investInFund(uint256) (contracts/core/IndexSwap.
    ↪ sol#170-226):
 External calls:
 - (tokenBalanceInBNB,vaultBalance) = indexSwapLibrary.
     ↪ getTokenAndVaultBalance(IIndexSwap(address(this))) (contracts/
     ↪ core/IndexSwap.sol#184-185)
 - investedAmountAfterSlippage = _swapETHToTokens(tokenAmount,amount,
     ↪ _slippage) (contracts/core/IndexSwap.sol#194-198)
  - swapResult = adapter._swapETHToToken{value: swapAmount}(t,swapAmount
      ↪ ,vault,_slippage) (contracts/core/IndexSwap.sol#349-354)
 External calls sending eth:
 - investedAmountAfterSlippage = _swapETHToTokens(tokenAmount,amount,
     ↪ _slippage) (contracts/core/IndexSwap.sol#194-198)
  - swapResult = adapter._swapETHToToken{value: swapAmount}(t,swapAmount
      ↪ ,vault,_slippage) (contracts/core/IndexSwap.sol#349-354)
 State variables written after the call(s):
 - _mint(msg.sender,tokenAmount) (contracts/core/IndexSwap.sol#217)
  - _balances[account] += amount (node_modules/@openzeppelin/contracts/
      ↪ token/ERC20/ERC20.sol#263)
```

```
Reentrancy in Rebalancing.rebalance(uint256) (contracts/rebalance/
    ↪ Rebalancing.sol#208-243):
 External calls:
 - (tokenBalanceInBNB,vaultBalance) = indexSwapLibrary.
     ↪ getTokenAndVaultBalance(index) (contracts/rebalance/Rebalancing.
     ↪ sol#223-224)
 - sumWeightsToSwap = sellTokens(oldWeights,newWeights,_slippage) (
     ↪ contracts/rebalance/Rebalancing.sol#235-239)
  - adapter._pullFromVault(index,index.getTokens()[i],swapAmount,address
      ↪ (this)) (contracts/rebalance/Rebalancing.sol#134-139)
  - adapter.redeemBNB(tokenMetadata.vTokens(index.getTokens()[i]),
      ↪ swapAmount,address(this)) (contracts/rebalance/Rebalancing.sol
      ↪ #145-149)
  - IWETH(index.getTokens()[i]).withdraw(swapAmount) (contracts/
      ↪ rebalance/Rebalancing.sol#151)
  - adapter._pullFromVault(index,index.getTokens()[i],swapAmount,address
      ↪ (adapter)) (contracts/rebalance/Rebalancing.sol#154-159)
  - adapter._swapTokenToETH(index.getTokens()[i],swapAmount,address(this
      ↪ ),_slippage) (contracts/rebalance/Rebalancing.sol#160-165)
 - buyTokens(oldWeights,newWeights,sumWeightsToSwap,_slippage) (
     ↪ contracts/rebalance/Rebalancing.sol#240)
  - adapter._swapETHToToken{value: swapAmount}(index.getTokens()[i],
      ↪ swapAmount,index.vault(),_slippage) (contracts/rebalance/
      ↪ Rebalancing.sol#195-200)
 External calls sending eth:
 - buyTokens(oldWeights,newWeights,sumWeightsToSwap,_slippage) (
     ↪ contracts/rebalance/Rebalancing.sol#240)
  - adapter._swapETHToToken{value: swapAmount}(index.getTokens()[i],
      ↪ swapAmount,index.vault(),_slippage) (contracts/rebalance/
      ↪ Rebalancing.sol#195-200)
 State variables written after the call(s):
 - lastRebalanced = block.timestamp (contracts/rebalance/Rebalancing.sol
     ↪ #242)
```

```
Reentrancy in Rebalancing.setPause(bool) (contracts/rebalance/
    ↪ Rebalancing.sol#84-106):
 External calls:
 - index.setPaused(_state) (contracts/rebalance/Rebalancing.sol#90)
 State variables written after the call(s):
 - lastPaused = block.timestamp (contracts/rebalance/Rebalancing.sol#91)
Reentrancy in Rebalancing.updateTokens(address[],uint96[],uint256) (
    ↪ contracts/rebalance/Rebalancing.sol#291-362):
 External calls:
 - index.updateRecords(tokens,denorms) (contracts/rebalance/Rebalancing.
     ↪ sol#355)
 - index.updateTokenList(tokens) (contracts/rebalance/Rebalancing.sol
     ↪ #357)
 - rebalance(_slippage) (contracts/rebalance/Rebalancing.sol#359)
  - (tokenBalanceInBNB,vaultBalance) = indexSwapLibrary.
      ↪ getTokenAndVaultBalance(index) (contracts/rebalance/Rebalancing.
      ↪ sol#223-224)
  - adapter._pullFromVault(index,index.getTokens()[i],swapAmount,address
      ↪ (this)) (contracts/rebalance/Rebalancing.sol#134-139)
  - adapter._swapETHToToken{value: swapAmount}(index.getTokens()[i],
      ↪ swapAmount,index.vault(),_slippage) (contracts/rebalance/
      ↪ Rebalancing.sol#195-200)
  - adapter.redeemBNB(tokenMetadata.vTokens(index.getTokens()[i]),
      ↪ swapAmount,address(this)) (contracts/rebalance/Rebalancing.sol
      ↪ #145-149)
  - IWETH(index.getTokens()[i]).withdraw(swapAmount) (contracts/
      ↪ rebalance/Rebalancing.sol#151)
  - adapter._pullFromVault(index,index.getTokens()[i],swapAmount,address
      ↪ (adapter)) (contracts/rebalance/Rebalancing.sol#154-159)
  - adapter._swapTokenToETH(index.getTokens()[i],swapAmount,address(this
      ↪ ),_slippage) (contracts/rebalance/Rebalancing.sol#160-165)
 External calls sending eth:
 - rebalance(_slippage) (contracts/rebalance/Rebalancing.sol#359)
```

```
  - adapter._swapETHToToken{value: swapAmount}(index.getTokens()[i],
      ↪ swapAmount,index.vault(),_slippage) (contracts/rebalance/
      ↪ Rebalancing.sol#195-200)
 State variables written after the call(s):
 - rebalance(_slippage) (contracts/rebalance/Rebalancing.sol#359)
  - lastRebalanced = block.timestamp (contracts/rebalance/Rebalancing.
      ↪ sol#242)
Reentrancy in Rebalancing.updateWeights(uint96[],uint256) (contracts/
    ↪ rebalance/Rebalancing.sol#249-262):
 External calls:
 - index.updateRecords(index.getTokens(),denorms) (contracts/rebalance/
     ↪ Rebalancing.sol#259)
 - rebalance(_slippage) (contracts/rebalance/Rebalancing.sol#260)
  - (tokenBalanceInBNB,vaultBalance) = indexSwapLibrary.
      ↪ getTokenAndVaultBalance(index) (contracts/rebalance/Rebalancing.
      ↪ sol#223-224)
  - adapter._pullFromVault(index,index.getTokens()[i],swapAmount,address
      ↪ (this)) (contracts/rebalance/Rebalancing.sol#134-139)
  - adapter._swapETHToToken{value: swapAmount}(index.getTokens()[i],
      ↪ swapAmount,index.vault(),_slippage) (contracts/rebalance/
      ↪ Rebalancing.sol#195-200)
  - adapter.redeemBNB(tokenMetadata.vTokens(index.getTokens()[i]),
      ↪ swapAmount,address(this)) (contracts/rebalance/Rebalancing.sol
      ↪ #145-149)
  - IWETH(index.getTokens()[i]).withdraw(swapAmount) (contracts/
      ↪ rebalance/Rebalancing.sol#151)
  - adapter._pullFromVault(index,index.getTokens()[i],swapAmount,address
      ↪ (adapter)) (contracts/rebalance/Rebalancing.sol#154-159)
  - adapter._swapTokenToETH(index.getTokens()[i],swapAmount,address(this
      ↪ ),_slippage) (contracts/rebalance/Rebalancing.sol#160-165)
 External calls sending eth:
 - rebalance(_slippage) (contracts/rebalance/Rebalancing.sol#260)
  - adapter._swapETHToToken{value: swapAmount}(index.getTokens()[i],
      ↪ swapAmount,index.vault(),_slippage) (contracts/rebalance/
```

```
    ↪ Rebalancing.sol#195-200)
 State variables written after the call(s):
 - rebalance(_slippage) (contracts/rebalance/Rebalancing.sol#260)
  - lastRebalanced = block.timestamp (contracts/rebalance/Rebalancing.
    ↪ sol#242)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
   ↪ #reentrancy-vulnerabilities-2

Reentrancy in IndexFactory.createIndex(string,string,uint256,uint256,
   ↪ bool) (contracts/IndexFactory.sol#64-166):
 External calls:
 - _adapter.init(address(accessController),uniswapRouter,address(_safe),
    ↪ tokenMetaDataInit) (contracts/IndexFactory.sol#85-90)
 Event emitted after the call(s):
 - IndexCreation(address(indexSwap),_name,_symbol,outAsset,address(_safe
    ↪ ),maxInvestmentAmount,address(_adapter),address(accessController
    ↪ )) (contracts/IndexFactory.sol#106-115)
Reentrancy in IndexFactory.createIndex(string,string,uint256,uint256,
   ↪ bool) (contracts/IndexFactory.sol#64-166):
 External calls:
 - _adapter.init(address(accessController),uniswapRouter,address(_safe),
    ↪ tokenMetaDataInit) (contracts/IndexFactory.sol#85-90)
 - rebalancing.init(IIndexSwap(address(indexSwap)),indexSwapLibrary,
    ↪ address(_adapter),address(accessController),tokenMetaDataInit) (
    ↪ contracts/IndexFactory.sol#121-127)
 Event emitted after the call(s):
 - IndexCreation(address(indexSwap),_name,_symbol,outAsset,address(_safe
    ↪ ),maxInvestmentAmount,address(_adapter),address(accessController
    ↪ )) (contracts/IndexFactory.sol#134-143)
 - RebalanceCreation(address(rebalancing)) (contracts/IndexFactory.sol
    ↪ #133)
Reentrancy in Rebalancing.feeModule() (contracts/rebalance/Rebalancing.
   ↪ sol#365-440):
 External calls:
```

```
- adapter._pullFromVault(index,index.getTokens()[i],amount,address(
    ↪ adapter)) (contracts/rebalance/Rebalancing.sol#384-389)
- adapter.redeemBNB(tokenMetadata.vTokens(index.getTokens()[i]),amount,
    ↪ index.treasury()) (contracts/rebalance/Rebalancing.sol#391-395)
- adapter._pullFromVault(index,index.getTokens()[i],amount,address(this
    ↪ )) (contracts/rebalance/Rebalancing.sol#397-402)
- IWETH(index.getTokens()[i]).withdraw(amount) (contracts/rebalance/
    ↪ Rebalancing.sol#404)
- (success) = address(index.treasury()).call{value: amount}() (
    ↪ contracts/rebalance/Rebalancing.sol#406-408)
- adapter._pullFromVault(index,index.getTokens()[i],amount,address(
    ↪ adapter)) (contracts/rebalance/Rebalancing.sol#413-418)
- adapter.redeemToken(tokenMetadata.vTokens(index.getTokens()[i]),index
    ↪ .getTokens()[i],amount,index.treasury()) (contracts/rebalance/
    ↪ Rebalancing.sol#420-425)
- adapter._pullFromVault(index,index.getTokens()[i],amount,index.
    ↪ treasury()) (contracts/rebalance/Rebalancing.sol#427-432)
External calls sending eth:
- (success) = address(index.treasury()).call{value: amount}() (
    ↪ contracts/rebalance/Rebalancing.sol#406-408)
Event emitted after the call(s):
- FeeCharged(block.timestamp,index.getTokens()[i],amount) (contracts/
    ↪ rebalance/Rebalancing.sol#436)
Reentrancy in IndexSwap.investInFund(uint256) (contracts/core/IndexSwap.
   ↪ sol#170-226):
External calls:
- (tokenBalanceInBNB,vaultBalance) = indexSwapLibrary.
    ↪ getTokenAndVaultBalance(IIndexSwap(address(this))) (contracts/
    ↪ core/IndexSwap.sol#184-185)
- investedAmountAfterSlippage = _swapETHToTokens(tokenAmount,amount,
    ↪ _slippage) (contracts/core/IndexSwap.sol#194-198)
 - swapResult = adapter._swapETHToToken{value: swapAmount}(t,swapAmount
     ↪ ,vault,_slippage) (contracts/core/IndexSwap.sol#349-354)
External calls sending eth:
```

```
 - investedAmountAfterSlippage = _swapETHToTokens(tokenAmount,amount,
    ↪ _slippage) (contracts/core/IndexSwap.sol#194-198)
  - swapResult = adapter._swapETHToToken{value: swapAmount}(t,swapAmount
     ↪ ,vault,_slippage) (contracts/core/IndexSwap.sol#349-354)
 Event emitted after the call(s):
 - InvestInFund(block.timestamp,msg.sender,msg.value,tokenAmount) (
    ↪ contracts/core/IndexSwap.sol#219)
 - Transfer(address(0),account,amount) (node_modules/@openzeppelin/
    ↪ contracts/token/ERC20/ERC20.sol#264)
  - _mint(msg.sender,tokenAmount) (contracts/core/IndexSwap.sol#217)
Reentrancy in Rebalancing.updateTokens(address[],uint96[],uint256) (
   ↪ contracts/rebalance/Rebalancing.sol#291-362):
 External calls:
 - index.updateRecords(tokens,denorms) (contracts/rebalance/Rebalancing.
    ↪ sol#355)
 - index.updateTokenList(tokens) (contracts/rebalance/Rebalancing.sol
    ↪ #357)
 - rebalance(_slippage) (contracts/rebalance/Rebalancing.sol#359)
  - (tokenBalanceInBNB,vaultBalance) = indexSwapLibrary.
     ↪ getTokenAndVaultBalance(index) (contracts/rebalance/Rebalancing.
     ↪ sol#223-224)
  - adapter._pullFromVault(index,index.getTokens()[i],swapAmount,address
     ↪ (this)) (contracts/rebalance/Rebalancing.sol#134-139)
  - adapter._swapETHToToken{value: swapAmount}(index.getTokens()[i],
     ↪ swapAmount,index.vault(),_slippage) (contracts/rebalance/
     ↪ Rebalancing.sol#195-200)
  - adapter.redeemBNB(tokenMetadata.vTokens(index.getTokens()[i]),
     ↪ swapAmount,address(this)) (contracts/rebalance/Rebalancing.sol
     ↪ #145-149)
  - IWETH(index.getTokens()[i]).withdraw(swapAmount) (contracts/
     ↪ rebalance/Rebalancing.sol#151)
  - adapter._pullFromVault(index,index.getTokens()[i],swapAmount,address
     ↪ (adapter)) (contracts/rebalance/Rebalancing.sol#154-159)
```

```
 - adapter._swapTokenToETH(index.getTokens()[i],swapAmount,address(this
    ↪ ),_slippage) (contracts/rebalance/Rebalancing.sol#160-165)
External calls sending eth:
- rebalance(_slippage) (contracts/rebalance/Rebalancing.sol#359)
 - adapter._swapETHToToken{value: swapAmount}(index.getTokens()[i],
    ↪ swapAmount,index.vault(),_slippage) (contracts/rebalance/
    ↪ Rebalancing.sol#195-200)
Event emitted after the call(s):
- UpdatedTokens(block.timestamp,tokens,denorms) (contracts/rebalance/
   ↪ Rebalancing.sol#361)
Reentrancy in Rebalancing.updateWeights(uint96[],uint256) (contracts/
   ↪ rebalance/Rebalancing.sol#249-262):
External calls:
- index.updateRecords(index.getTokens(),denorms) (contracts/rebalance/
   ↪ Rebalancing.sol#259)
- rebalance(_slippage) (contracts/rebalance/Rebalancing.sol#260)
 - (tokenBalanceInBNB,vaultBalance) = indexSwapLibrary.
    ↪ getTokenAndVaultBalance(index) (contracts/rebalance/Rebalancing.
    ↪ sol#223-224)
 - adapter._pullFromVault(index,index.getTokens()[i],swapAmount,address
    ↪ (this)) (contracts/rebalance/Rebalancing.sol#134-139)
 - adapter._swapETHToToken{value: swapAmount}(index.getTokens()[i],
    ↪ swapAmount,index.vault(),_slippage) (contracts/rebalance/
    ↪ Rebalancing.sol#195-200)
 - adapter.redeemBNB(tokenMetadata.vTokens(index.getTokens()[i]),
    ↪ swapAmount,address(this)) (contracts/rebalance/Rebalancing.sol
    ↪ #145-149)
 - IWETH(index.getTokens()[i]).withdraw(swapAmount) (contracts/
    ↪ rebalance/Rebalancing.sol#151)
 - adapter._pullFromVault(index,index.getTokens()[i],swapAmount,address
    ↪ (adapter)) (contracts/rebalance/Rebalancing.sol#154-159)
 - adapter._swapTokenToETH(index.getTokens()[i],swapAmount,address(this
    ↪ ),_slippage) (contracts/rebalance/Rebalancing.sol#160-165)
External calls sending eth:
```

```
 - rebalance(_slippage) (contracts/rebalance/Rebalancing.sol#260)
  - adapter._swapETHToToken{value: swapAmount}(index.getTokens()[i],
     ↪ swapAmount,index.vault(),_slippage) (contracts/rebalance/
     ↪ Rebalancing.sol#195-200)
 Event emitted after the call(s):
 - UpdatedWeights(block.timestamp,denorms) (contracts/rebalance/
    ↪ Rebalancing.sol#261)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
   ↪ #reentrancy-vulnerabilities-3

Adapter.lendToken(address,address,uint256,address) (contracts/core/
   ↪ Adapter.sol#219-236) uses timestamp for comparisons
 Dangerous comparisons:
 - assert(bool)(vToken.mint(_amount) == 0) (contracts/core/Adapter.sol
    ↪ #233)
IndexSwap._swapETHToTokens(uint256,uint256[],uint256) (contracts/core/
   ↪ IndexSwap.sol#332-360) uses timestamp for comparisons
 Dangerous comparisons:
 - require(bool,string)(address(this).balance >= swapAmount,not enough
    ↪ bnb) (contracts/core/IndexSwap.sol#347)
Rebalancing.setPause(bool) (contracts/rebalance/Rebalancing.sol#84-106)
   ↪ uses timestamp for comparisons
 Dangerous comparisons:
 - lastRebalanced > lastPaused || block.timestamp >= (lastPaused + 600)
    ↪ (contracts/rebalance/Rebalancing.sol#94-95)
Rebalancing.feeModule() (contracts/rebalance/Rebalancing.sol#365-440)
   ↪ uses timestamp for comparisons
 Dangerous comparisons:
 - require(bool,string)(lastFeeCharged < lastRebalanced,Fee has already
    ↪ been charged after the last rebalancing!) (contracts/rebalance/
    ↪ Rebalancing.sol#366-369)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
   ↪ #block-timestamp
```

```
AddressUpgradeable.verifyCallResult(bool,bytes,string) (node_modules/
    ↪ @openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol
    ↪ #174-194) uses assembly
 - INLINE ASM (node_modules/@openzeppelin/contracts-upgradeable/utils/
     ↪ AddressUpgradeable.sol#186-189)
Clones.clone(address) (node_modules/@openzeppelin/contracts/proxy/Clones
    ↪ .sol#25-34) uses assembly
 - INLINE ASM (node_modules/@openzeppelin/contracts/proxy/Clones.sol
     ↪ #26-32)
Clones.cloneDeterministic(address,bytes32) (node_modules/@openzeppelin/
    ↪ contracts/proxy/Clones.sol#43-52) uses assembly
 - INLINE ASM (node_modules/@openzeppelin/contracts/proxy/Clones.sol
     ↪ #44-50)
Clones.predictDeterministicAddress(address,bytes32,address) (
    ↪ node_modules/@openzeppelin/contracts/proxy/Clones.sol#57-72) uses
    ↪  assembly
 - INLINE ASM (node_modules/@openzeppelin/contracts/proxy/Clones.sol
     ↪ #62-71)
Address.verifyCallResult(bool,bytes,string) (node_modules/@openzeppelin/
    ↪ contracts/utils/Address.sol#201-221) uses assembly
 - INLINE ASM (node_modules/@openzeppelin/contracts/utils/Address.sol
     ↪ #213-216)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #assembly-usage

Different versions of Solidity is used:
 - Version used: ['>=0.6.0', '>=0.7.0<0.9.0', '^0.8.0', '^0.8.1',
     ↪ '^0.8.2', '^0.8.6']
 - ^0.8.0 (node_modules/@chainlink/contracts/src/v0.8/Denominations.sol
     ↪ #3)
 - ^0.8.0 (node_modules/@chainlink/contracts/src/v0.8/interfaces/
     ↪ AggregatorInterface.sol#2)
 - ^0.8.0 (node_modules/@chainlink/contracts/src/v0.8/interfaces/
     ↪ AggregatorV2V3Interface.sol#2)
```

- ^0.8.0 (node_modules/@chainlink/contracts/src/v0.8/interfaces/
  ↪ AggregatorV3Interface.sol#2)
- >=0.7.0<0.9.0 (node_modules/@gnosis.pm/safe-contracts/contracts/
  ↪ common/Enum.sol#2)
- >=0.7.0<0.9.0 (node_modules/@gnosis.pm/zodiac/contracts/core/Module.
  ↪ sol#4)
- >=0.7.0<0.9.0 (node_modules/@gnosis.pm/zodiac/contracts/factory/
  ↪ FactoryFriendly.sol#4)
- >=0.7.0<0.9.0 (node_modules/@gnosis.pm/zodiac/contracts/guard/
  ↪ BaseGuard.sol#2)
- >=0.7.0<0.9.0 (node_modules/@gnosis.pm/zodiac/contracts/guard/
  ↪ Guardable.sol#2)
- >=0.7.0<0.9.0 (node_modules/@gnosis.pm/zodiac/contracts/interfaces/
  ↪ IAvatar.sol#4)
- >=0.7.0<0.9.0 (node_modules/@gnosis.pm/zodiac/contracts/interfaces/
  ↪ IGuard.sol#2)
- ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/access/
  ↪ OwnableUpgradeable.sol#4)
- ^0.8.2 (node_modules/@openzeppelin/contracts-upgradeable/proxy/utils/
  ↪ Initializable.sol#4)
- ^0.8.1 (node_modules/@openzeppelin/contracts-upgradeable/utils/
  ↪ AddressUpgradeable.sol#4)
- ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/utils/
  ↪ ContextUpgradeable.sol#4)
- ^0.8.0 (node_modules/@openzeppelin/contracts/access/AccessControl.sol
  ↪ #4)
- ^0.8.0 (node_modules/@openzeppelin/contracts/access/IAccessControl.
  ↪ sol#4)
- ^0.8.0 (node_modules/@openzeppelin/contracts/access/Ownable.sol#4)
- ^0.8.0 (node_modules/@openzeppelin/contracts/proxy/Clones.sol#4)
- ^0.8.2 (node_modules/@openzeppelin/contracts/proxy/utils/
  ↪ Initializable.sol#4)
- ^0.8.0 (node_modules/@openzeppelin/contracts/security/ReentrancyGuard
  ↪ .sol#4)

```
- ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#4)
- ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/IERC20.sol
   ↪ #4)
- ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/extensions/
   ↪ ERC20Burnable.sol#4)
- ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/extensions/
   ↪ IERC20Metadata.sol#4)
- ^0.8.1 (node_modules/@openzeppelin/contracts/utils/Address.sol#4)
- ^0.8.0 (node_modules/@openzeppelin/contracts/utils/Context.sol#4)
- ^0.8.0 (node_modules/@openzeppelin/contracts/utils/Strings.sol#4)
- ^0.8.0 (node_modules/@openzeppelin/contracts/utils/introspection/
   ↪ ERC165.sol#4)
- ^0.8.0 (node_modules/@openzeppelin/contracts/utils/introspection/
   ↪ IERC165.sol#4)
- ^0.8.0 (node_modules/@openzeppelin/contracts/utils/math/SafeMath.sol
   ↪ #4)
- >=0.6.0 (node_modules/@uniswap/lib/contracts/libraries/TransferHelper
   ↪ .sol#3)
- ^0.8.6 (contracts/IndexFactory.sol#2)
- ^0.8.6 (contracts/access/AccessController.sol#12)
- ^0.8.6 (contracts/core/Adapter.sol#15)
- ^0.8.6 (contracts/core/IndexSwap.sol#12)
- ^0.8.6 (contracts/core/IndexSwapLibrary.sol#12)
- ^0.8.6 (contracts/interfaces/IAccessController.sol#12)
- ^0.8.6 (contracts/interfaces/IAdapter.sol#15)
- ^0.8.6 (contracts/interfaces/IIndexSwap.sol#12)
- ^0.8.6 (contracts/interfaces/IIndexSwapLibrary.sol#11)
- ^0.8.6 (contracts/interfaces/IPriceOracle.sol#2)
- ^0.8.6 (contracts/interfaces/IRebalancing.sol#15)
- ^0.8.6 (contracts/interfaces/IUniswapV2Router02.sol#2)
- ^0.8.6 (contracts/interfaces/IVault.sol#11)
- ^0.8.6 (contracts/interfaces/IWETH.sol#2)
- ^0.8.6 (contracts/mock/ERC20Mock.sol#2)
- ^0.8.6 (contracts/oracle/PriceOracle.sol#2)
```

```
 - ^0.8.6 (contracts/rebalance/Rebalancing.sol#15)
 - ^0.8.6 (contracts/vault/Vault.sol#3)
 - ^0.8.6 (contracts/vault/VelvetSafeModule.sol#11)
 - ^0.8.6 (contracts/venus/ComptrollerInterface.sol#2)
 - ^0.8.6 (contracts/venus/IVBNB.sol#2)
 - ^0.8.6 (contracts/venus/TokenMetadata.sol#10)
 - ^0.8.6 (contracts/venus/VBep20Interface.sol#2)
 - ^0.8.6 (contracts/venus/VBep20Storage.sol#2)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #different-pragma-directives-are-used

solc-0.8.10 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #incorrect-versions-of-solidity

Pragma version^0.8.0 (node_modules/@chainlink/contracts/src/v0.8/
    ↪ Denominations.sol#3) allows old versions
Pragma version^0.8.0 (node_modules/@chainlink/contracts/src/v0.8/
    ↪ interfaces/AggregatorInterface.sol#2) allows old versions
Pragma version^0.8.0 (node_modules/@chainlink/contracts/src/v0.8/
    ↪ interfaces/AggregatorV2V3Interface.sol#2) allows old versions
Pragma version^0.8.0 (node_modules/@chainlink/contracts/src/v0.8/
    ↪ interfaces/AggregatorV3Interface.sol#2) allows old versions
Pragma version>=0.7.0<0.9.0 (node_modules/@gnosis.pm/safe-contracts/
    ↪ contracts/common/Enum.sol#2) is too complex
Pragma version>=0.7.0<0.9.0 (node_modules/@gnosis.pm/zodiac/contracts/
    ↪ core/Module.sol#4) is too complex
Pragma version>=0.7.0<0.9.0 (node_modules/@gnosis.pm/zodiac/contracts/
    ↪ factory/FactoryFriendly.sol#4) is too complex
Pragma version>=0.7.0<0.9.0 (node_modules/@gnosis.pm/zodiac/contracts/
    ↪ guard/BaseGuard.sol#2) is too complex
Pragma version>=0.7.0<0.9.0 (node_modules/@gnosis.pm/zodiac/contracts/
    ↪ guard/Guardable.sol#2) is too complex
```

```
Pragma version>=0.7.0<0.9.0 (node_modules/@gnosis.pm/zodiac/contracts/
   ↪ interfaces/IAvatar.sol#4) is too complex
Pragma version>=0.7.0<0.9.0 (node_modules/@gnosis.pm/zodiac/contracts/
   ↪ interfaces/IGuard.sol#2) is too complex
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/
   ↪ access/OwnableUpgradeable.sol#4) allows old versions
Pragma version^0.8.2 (node_modules/@openzeppelin/contracts-upgradeable/
   ↪ proxy/utils/Initializable.sol#4) allows old versions
Pragma version^0.8.1 (node_modules/@openzeppelin/contracts-upgradeable/
   ↪ utils/AddressUpgradeable.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/
   ↪ utils/ContextUpgradeable.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/access/
   ↪ AccessControl.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/access/
   ↪ IAccessControl.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/access/
   ↪ Ownable.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/proxy/Clones.
   ↪ sol#4) allows old versions
Pragma version^0.8.2 (node_modules/@openzeppelin/contracts/proxy/utils/
   ↪ Initializable.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/security/
   ↪ ReentrancyGuard.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/
   ↪ ERC20.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/
   ↪ IERC20.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/
   ↪ extensions/ERC20Burnable.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/
   ↪ extensions/IERC20Metadata.sol#4) allows old versions
Pragma version^0.8.1 (node_modules/@openzeppelin/contracts/utils/Address
   ↪ .sol#4) allows old versions
```

```
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/utils/Context
   ↪ .sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/utils/Strings
   ↪ .sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/utils/
   ↪ introspection/ERC165.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/utils/
   ↪ introspection/IERC165.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/utils/math/
   ↪ SafeMath.sol#4) allows old versions
Pragma version>=0.6.0 (node_modules/@uniswap/lib/contracts/libraries/
   ↪ TransferHelper.sol#3) allows old versions
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#60-65):
 - (success) = recipient.call{value: amount}() (node_modules/
    ↪ @openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol
    ↪ #63)
Low level call in AddressUpgradeable.functionCallWithValue(address,bytes
   ↪ ,uint256,string) (node_modules/@openzeppelin/contracts-
   ↪ upgradeable/utils/AddressUpgradeable.sol#128-139):
 - (success,returndata) = target.call{value: value}(data) (node_modules/
    ↪ @openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol
    ↪ #137)
Low level call in AddressUpgradeable.functionStaticCall(address,bytes,
   ↪ string) (node_modules/@openzeppelin/contracts-upgradeable/utils/
   ↪ AddressUpgradeable.sol#157-166):
 - (success,returndata) = target.staticcall(data) (node_modules/
    ↪ @openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol
    ↪ #164)
```

```
Low level call in Address.sendValue(address,uint256) (node_modules/
    ↪ @openzeppelin/contracts/utils/Address.sol#60-65):
 - (success) = recipient.call{value: amount}() (node_modules/
     ↪ @openzeppelin/contracts/utils/Address.sol#63)
Low level call in Address.functionCallWithValue(address,bytes,uint256,
    ↪ string) (node_modules/@openzeppelin/contracts/utils/Address.sol
    ↪ #128-139):
 - (success,returndata) = target.call{value: value}(data) (node_modules/
     ↪ @openzeppelin/contracts/utils/Address.sol#137)
Low level call in Address.functionStaticCall(address,bytes,string) (
    ↪ node_modules/@openzeppelin/contracts/utils/Address.sol#157-166):
 - (success,returndata) = target.staticcall(data) (node_modules/
     ↪ @openzeppelin/contracts/utils/Address.sol#164)
Low level call in Address.functionDelegateCall(address,bytes,string) (
    ↪ node_modules/@openzeppelin/contracts/utils/Address.sol#184-193):
 - (success,returndata) = target.delegatecall(data) (node_modules/
     ↪ @openzeppelin/contracts/utils/Address.sol#191)
Low level call in TransferHelper.safeApprove(address,address,uint256) (
    ↪ node_modules/@uniswap/lib/contracts/libraries/TransferHelper.sol
    ↪ #7-18):
 - (success,data) = token.call(abi.encodeWithSelector(0x095ea7b3,to,
     ↪ value)) (node_modules/@uniswap/lib/contracts/libraries/
     ↪ TransferHelper.sol#13)
Low level call in TransferHelper.safeTransfer(address,address,uint256) (
    ↪ node_modules/@uniswap/lib/contracts/libraries/TransferHelper.sol
    ↪ #20-31):
 - (success,data) = token.call(abi.encodeWithSelector(0xa9059cbb,to,
     ↪ value)) (node_modules/@uniswap/lib/contracts/libraries/
     ↪ TransferHelper.sol#26)
Low level call in TransferHelper.safeTransferFrom(address,address,
    ↪ address,uint256) (node_modules/@uniswap/lib/contracts/libraries/
    ↪ TransferHelper.sol#33-45):
 - (success,data) = token.call(abi.encodeWithSelector(0x23b872dd,from,to
     ↪ ,value)) (node_modules/@uniswap/lib/contracts/libraries/
```

```
    ↪ TransferHelper.sol#40)
Low level call in TransferHelper.safeTransferETH(address,uint256) (
   ↪ node_modules/@uniswap/lib/contracts/libraries/TransferHelper.sol
   ↪ #47-50):
 - (success) = to.call{value: value}(new bytes(0)) (node_modules/
    ↪ @uniswap/lib/contracts/libraries/TransferHelper.sol#48)
Low level call in Adapter._swapTokenToETH(address,uint256,address,
   ↪ uint256) (contracts/core/Adapter.sol#158-216):
 - (success) = address(to).call{value: swapResult}() (contracts/core/
    ↪ Adapter.sol#169)
 - (success) = address(to).call{value: swapAmount}() (contracts/core/
    ↪ Adapter.sol#203)
Low level call in Adapter.redeemBNB(address,uint256,address) (contracts/
   ↪ core/Adapter.sol#277-296):
 - (success) = address(_to).call{value: address(this).balance}() (
    ↪ contracts/core/Adapter.sol#291-293)
Low level call in IndexSwap.investInFund(uint256) (contracts/core/
   ↪ IndexSwap.sol#170-226):
 - (success) = address(msg.sender).call{value: address(this).balance}()
    ↪ (contracts/core/IndexSwap.sol#222-224)
Low level call in IndexSwap.withdrawFund(uint256,uint256,bool) (
   ↪ contracts/core/IndexSwap.sol#233-324):
 - (success) = address(msg.sender).call{value: amount}() (contracts/core
    ↪ /IndexSwap.sol#288-290)
Low level call in Rebalancing.feeModule() (contracts/rebalance/
   ↪ Rebalancing.sol#365-440):
 - (success) = address(index.treasury()).call{value: amount}() (
    ↪ contracts/rebalance/Rebalancing.sol#406-408)
Low level call in Vault.executeTransactionETH(address,uint256) (
   ↪ contracts/vault/Vault.sol#27-34):
 - (successReturn) = address(_to).call{value: value}() (contracts/vault/
    ↪ Vault.sol#32)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
   ↪ #low-level-calls
```

```
BaseGuard (node_modules/@gnosis.pm/zodiac/contracts/guard/BaseGuard.sol
    ↪ #8-38) should inherit from IGuard (node_modules/@gnosis.pm/zodiac
    ↪ /contracts/interfaces/IGuard.sol#6-22)
AccessController (contracts/access/AccessController.sol#16-79) should
    ↪ inherit from IAccessController (contracts/interfaces/
    ↪ IAccessController.sol#14-22)
Adapter (contracts/core/Adapter.sol#35-349) should inherit from IAdapter
    ↪  (contracts/interfaces/IAdapter.sol#18-97)
IndexSwapLibrary (contracts/core/IndexSwapLibrary.sol#22-177) should
    ↪ inherit from IIndexSwapLibrary (contracts/interfaces/
    ↪ IIndexSwapLibrary.sol#14-67)
Rebalancing (contracts/rebalance/Rebalancing.sol#33-448) should inherit
    ↪ from IRebalancing (contracts/interfaces/IRebalancing.sol#21-66)
Vault (contracts/vault/Vault.sol#7-43) should inherit from IVault (
    ↪ contracts/interfaces/IVault.sol#15-25)
VelvetSafeModule (contracts/vault/VelvetSafeModule.sol#19-73) should
    ↪ inherit from IVault (contracts/interfaces/IVault.sol#15-25)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #missing-inheritance

Function IUniswapV2Router.WETH() (contracts/interfaces/IUniswapV2Router.
    ↪ sol#7) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #conformance-to-solidity-naming-conventions

Parameter Module.setAvatar(address)._avatar (node_modules/@gnosis.pm/
    ↪ zodiac/contracts/core/Module.sol#23) is not in mixedCase
Parameter Module.setTarget(address)._target (node_modules/@gnosis.pm/
    ↪ zodiac/contracts/core/Module.sol#31) is not in mixedCase
Parameter Guardable.setGuard(address)._guard (node_modules/@gnosis.pm/
    ↪ zodiac/contracts/guard/Guardable.sol#19) is not in mixedCase
Function OwnableUpgradeable.__Ownable_init() (node_modules/@openzeppelin
    ↪ /contracts-upgradeable/access/OwnableUpgradeable.sol#29-31) is
```

```
    ↪ not in mixedCase
Function OwnableUpgradeable.__Ownable_init_unchained() (node_modules/
    ↪ @openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol
    ↪ #33-35) is not in mixedCase
Variable OwnableUpgradeable.__gap (node_modules/@openzeppelin/contracts-
    ↪ upgradeable/access/OwnableUpgradeable.sol#87) is not in mixedCase
Function ContextUpgradeable.__Context_init() (node_modules/@openzeppelin
    ↪ /contracts-upgradeable/utils/ContextUpgradeable.sol#18-19) is not
    ↪  in mixedCase
Function ContextUpgradeable.__Context_init_unchained() (node_modules/
    ↪ @openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol
    ↪ #21-22) is not in mixedCase
Variable ContextUpgradeable.__gap (node_modules/@openzeppelin/contracts-
    ↪ upgradeable/utils/ContextUpgradeable.sol#36) is not in mixedCase
Parameter IndexFactory.createIndex(string,string,uint256,uint256,bool).
    ↪ _name (contracts/IndexFactory.sol#65) is not in mixedCase
Parameter IndexFactory.createIndex(string,string,uint256,uint256,bool).
    ↪ _symbol (contracts/IndexFactory.sol#66) is not in mixedCase
Parameter IndexFactory.createIndex(string,string,uint256,uint256,bool).
    ↪ _maxInvestmentAmount (contracts/IndexFactory.sol#67) is not in
    ↪ mixedCase
Parameter IndexFactory.createIndex(string,string,uint256,uint256,bool).
    ↪ _feePointBasis (contracts/IndexFactory.sol#68) is not in
    ↪ mixedCase
Parameter IndexFactory.initializeTokens(uint256,address[],uint96[]).
    ↪ _tokens (contracts/IndexFactory.sol#180) is not in mixedCase
Parameter IndexFactory.initializeTokens(uint256,address[],uint96[]).
    ↪ _denorms (contracts/IndexFactory.sol#181) is not in mixedCase
Parameter IndexFactory.setIndexSwapLibrary(address)._indexSwapLibrary (
    ↪ contracts/IndexFactory.sol#190) is not in mixedCase
Parameter IndexFactory.setOutAsset(address)._outAsset (contracts/
    ↪ IndexFactory.sol#195) is not in mixedCase
Variable IndexFactory.IndexSwapInfolList (contracts/IndexFactory.sol#28)
    ↪  is not in mixedCase
```

```
Parameter Adapter.init(address,address,address,address).
    ↪ _accessController (contracts/core/Adapter.sol#47) is not in
    ↪ mixedCase
Parameter Adapter.init(address,address,address,address).
    ↪ _pancakeSwapAddress (contracts/core/Adapter.sol#48) is not in
    ↪ mixedCase
Parameter Adapter.init(address,address,address,address)._safe (contracts
    ↪ /core/Adapter.sol#49) is not in mixedCase
Parameter Adapter.init(address,address,address,address)._tokenMetadata (
    ↪ contracts/core/Adapter.sol#50) is not in mixedCase
Function Adapter._pullFromVault(IIndexSwap,address,uint256,address) (
    ↪ contracts/core/Adapter.sol#76-99) is not in mixedCase
Parameter Adapter._pullFromVault(IIndexSwap,address,uint256,address).
    ↪ _index (contracts/core/Adapter.sol#77) is not in mixedCase
Function Adapter._swapETHToToken(address,uint256,address,uint256) (
    ↪ contracts/core/Adapter.sol#108-148) is not in mixedCase
Parameter Adapter._swapETHToToken(address,uint256,address,uint256).
    ↪ _slippage (contracts/core/Adapter.sol#112) is not in mixedCase
Function Adapter._swapTokenToETH(address,uint256,address,uint256) (
    ↪ contracts/core/Adapter.sol#158-216) is not in mixedCase
Parameter Adapter._swapTokenToETH(address,uint256,address,uint256).
    ↪ _slippage (contracts/core/Adapter.sol#162) is not in mixedCase
Parameter Adapter.lendToken(address,address,uint256,address).
    ↪ _underlyingAsset (contracts/core/Adapter.sol#220) is not in
    ↪ mixedCase
Parameter Adapter.lendToken(address,address,uint256,address)._vAsset (
    ↪ contracts/core/Adapter.sol#221) is not in mixedCase
Parameter Adapter.lendToken(address,address,uint256,address)._amount (
    ↪ contracts/core/Adapter.sol#222) is not in mixedCase
Parameter Adapter.lendToken(address,address,uint256,address)._to (
    ↪ contracts/core/Adapter.sol#223) is not in mixedCase
Parameter Adapter.lendBNB(address,address,uint256,address).
    ↪ _underlyingAsset (contracts/core/Adapter.sol#239) is not in
    ↪ mixedCase
```

```
Parameter Adapter.lendBNB(address,address,uint256,address)._vAsset (
    ↪ contracts/core/Adapter.sol#240) is not in mixedCase
Parameter Adapter.lendBNB(address,address,uint256,address)._amount (
    ↪ contracts/core/Adapter.sol#241) is not in mixedCase
Parameter Adapter.lendBNB(address,address,uint256,address)._to (
    ↪ contracts/core/Adapter.sol#242) is not in mixedCase
Parameter Adapter.redeemToken(address,address,uint256,address)._vAsset (
    ↪ contracts/core/Adapter.sol#258) is not in mixedCase
Parameter Adapter.redeemToken(address,address,uint256,address).
    ↪ _underlying (contracts/core/Adapter.sol#259) is not in mixedCase
Parameter Adapter.redeemToken(address,address,uint256,address)._amount (
    ↪ contracts/core/Adapter.sol#260) is not in mixedCase
Parameter Adapter.redeemToken(address,address,uint256,address)._to (
    ↪ contracts/core/Adapter.sol#261) is not in mixedCase
Parameter Adapter.redeemBNB(address,uint256,address)._vAsset (contracts/
    ↪ core/Adapter.sol#278) is not in mixedCase
Parameter Adapter.redeemBNB(address,uint256,address)._amount (contracts/
    ↪ core/Adapter.sol#279) is not in mixedCase
Parameter Adapter.redeemBNB(address,uint256,address)._to (contracts/core
    ↪ /Adapter.sol#280) is not in mixedCase
Parameter Adapter.getSlippage(uint256,uint256,address[])._amount (
    ↪ contracts/core/Adapter.sol#331) is not in mixedCase
Parameter Adapter.getSlippage(uint256,uint256,address[])._slippage (
    ↪ contracts/core/Adapter.sol#332) is not in mixedCase
Constant Adapter.divisor_int (contracts/core/Adapter.sol#44) is not in
    ↪ UPPER_CASE_WITH_UNDERSCORES
Event IndexSwapLOG_PUBLIC_SWAP_ENABLED() (contracts/core/IndexSwap.sol
    ↪ #115) is not in CapWords
Parameter IndexSwap.investInFund(uint256)._slippage (contracts/core/
    ↪ IndexSwap.sol#170) is not in mixedCase
Parameter IndexSwap.withdrawFund(uint256,uint256,bool)._slippage (
    ↪ contracts/core/IndexSwap.sol#233) is not in mixedCase
Parameter IndexSwap.setPaused(bool)._state (contracts/core/IndexSwap.sol
    ↪ #374) is not in mixedCase
```

```
Parameter IndexSwap.getRecord(address)._token (contracts/core/IndexSwap.
    ↪ sol#405) is not in mixedCase
Parameter IndexSwap.updateTreasury(address)._newTreasury (contracts/core
    ↪ /IndexSwap.sol#420) is not in mixedCase
Variable IndexSwap._tokens (contracts/core/IndexSwap.sol#60) is not in
    ↪ mixedCase
Variable IndexSwap._records (contracts/core/IndexSwap.sol#63) is not in
    ↪ mixedCase
Variable IndexSwap.MAX_INVESTMENTAMOUNT (contracts/core/IndexSwap.sol
    ↪ #69) is not in mixedCase
Parameter IndexSwapLibrary.getTokenAndVaultBalance(IIndexSwap)._index (
    ↪ contracts/core/IndexSwapLibrary.sol#45) is not in mixedCase
Parameter IndexSwapLibrary.getTokenBalance(IIndexSwap,address,bool).
    ↪ _index (contracts/core/IndexSwapLibrary.sol#112) is not in
    ↪ mixedCase
Parameter IndexSwapLibrary.calculateSwapAmounts(IIndexSwap,uint256,
    ↪ uint256[],uint256)._index (contracts/core/IndexSwapLibrary.sol
    ↪ #140) is not in mixedCase
Function IndexSwapLibrary._getTokenAmountInUSD(address,uint256) (
    ↪ contracts/core/IndexSwapLibrary.sol#162-168) is not in mixedCase
Function IndexSwapLibrary._getTokenPriceUSDETH(uint256) (contracts/core/
    ↪ IndexSwapLibrary.sol#170-176) is not in mixedCase
Function IAdapter._pullFromVault(IIndexSwap,address,uint256,address) (
    ↪ contracts/interfaces/IAdapter.sol#31-36) is not in mixedCase
Function IAdapter._swapETHToToken(address,uint256,address,uint256) (
    ↪ contracts/interfaces/IAdapter.sol#45-50) is not in mixedCase
Function IAdapter._swapTokenToETH(address,uint256,address,uint256) (
    ↪ contracts/interfaces/IAdapter.sol#60-65) is not in mixedCase
Event IIndexSwapLOG_PUBLIC_SWAP_ENABLED() (contracts/interfaces/
    ↪ IIndexSwap.sol#120) is not in CapWords
Function IIndexSwap.TOTAL_WEIGHT() (contracts/interfaces/IIndexSwap.sol
    ↪ #21) is not in mixedCase
Function IIndexSwapLibrary._getTokenAmountInUSD(address,uint256) (
    ↪ contracts/interfaces/IIndexSwapLibrary.sol#58-61) is not in
```

```
    ↪ mixedCase
Function IIndexSwapLibrary._getTokenPriceUSDETH(uint256) (contracts/
    ↪ interfaces/IIndexSwapLibrary.sol#63-66) is not in mixedCase
Function IPriceOracle._addFeed(address,address,AggregatorV2V3Interface)
    ↪ (contracts/interfaces/IPriceOracle.sol#7-11) is not in mixedCase
Function IUniswapV2Router02.WETH() (contracts/interfaces/
    ↪ IUniswapV2Router02.sol#7) is not in mixedCase
Function PriceOracle._addFeed(address,address,AggregatorV2V3Interface) (
    ↪ contracts/oracle/PriceOracle.sol#41-52) is not in mixedCase
Function PriceOracle._updateFeed(address,address,AggregatorV2V3Interface
    ↪ ) (contracts/oracle/PriceOracle.sol#60-66) is not in mixedCase
Parameter PriceOracle.getPriceTokenUSD(address,uint256)._base (contracts
    ↪ /oracle/PriceOracle.sol#143) is not in mixedCase
Parameter Rebalancing.init(IIndexSwap,address,address,address,address).
    ↪ _index (contracts/rebalance/Rebalancing.sol#59) is not in
    ↪ mixedCase
Parameter Rebalancing.init(IIndexSwap,address,address,address,address).
    ↪ _indexSwapLibrary (contracts/rebalance/Rebalancing.sol#60) is not
    ↪  in mixedCase
Parameter Rebalancing.init(IIndexSwap,address,address,address,address).
    ↪ _adapter (contracts/rebalance/Rebalancing.sol#61) is not in
    ↪ mixedCase
Parameter Rebalancing.init(IIndexSwap,address,address,address,address).
    ↪ _accessController (contracts/rebalance/Rebalancing.sol#62) is not
    ↪  in mixedCase
Parameter Rebalancing.init(IIndexSwap,address,address,address,address).
    ↪ _tokenMetadata (contracts/rebalance/Rebalancing.sol#63) is not in
    ↪  mixedCase
Parameter Rebalancing.setPause(bool)._state (contracts/rebalance/
    ↪ Rebalancing.sol#84) is not in mixedCase
Parameter Rebalancing.sellTokens(uint256[],uint256[],uint256).
    ↪ _oldWeights (contracts/rebalance/Rebalancing.sol#115) is not in
    ↪ mixedCase
```

```
Parameter Rebalancing.sellTokens(uint256[],uint256[],uint256).
    ↪ _newWeights (contracts/rebalance/Rebalancing.sol#116) is not in
    ↪ mixedCase
Parameter Rebalancing.sellTokens(uint256[],uint256[],uint256)._slippage
    ↪ (contracts/rebalance/Rebalancing.sol#117) is not in mixedCase
Parameter Rebalancing.buyTokens(uint256[],uint256[],uint256,uint256).
    ↪ _oldWeights (contracts/rebalance/Rebalancing.sol#180) is not in
    ↪ mixedCase
Parameter Rebalancing.buyTokens(uint256[],uint256[],uint256,uint256).
    ↪ _newWeights (contracts/rebalance/Rebalancing.sol#181) is not in
    ↪ mixedCase
Parameter Rebalancing.buyTokens(uint256[],uint256[],uint256,uint256).
    ↪ _slippage (contracts/rebalance/Rebalancing.sol#183) is not in
    ↪ mixedCase
Parameter Rebalancing.rebalance(uint256)._slippage (contracts/rebalance/
    ↪ Rebalancing.sol#208) is not in mixedCase
Parameter Rebalancing.updateWeights(uint96[],uint256)._slippage (
    ↪ contracts/rebalance/Rebalancing.sol#249) is not in mixedCase
Parameter Rebalancing.updateTokens(address[],uint96[],uint256)._slippage
    ↪  (contracts/rebalance/Rebalancing.sol#294) is not in mixedCase
Parameter Rebalancing.updateTreasury(address)._newAddress (contracts/
    ↪ rebalance/Rebalancing.sol#442) is not in mixedCase
Parameter Vault.executeTransactionETH(address,uint256)._to (contracts/
    ↪ vault/Vault.sol#27) is not in mixedCase
Parameter Vault.executeTransactionOther(address,uint256,address)._to (
    ↪ contracts/vault/Vault.sol#37) is not in mixedCase
Parameter Vault.executeTransactionOther(address,uint256,address)._token
    ↪ (contracts/vault/Vault.sol#39) is not in mixedCase
Parameter VelvetSafeModule.executeTransactionETH(address,uint256)._to (
    ↪ contracts/vault/VelvetSafeModule.sol#52) is not in mixedCase
Parameter VelvetSafeModule.executeTransactionOther(address,uint256,
    ↪ address)._to (contracts/vault/VelvetSafeModule.sol#61) is not in
    ↪ mixedCase
```

```
Parameter VelvetSafeModule.executeTransactionOther(address,uint256,
    ↪ address)._token (contracts/vault/VelvetSafeModule.sol#63) is not
    ↪ in mixedCase
Parameter TokenMetadata.add(address,address)._underlying (contracts/
    ↪ venus/TokenMetadata.sol#19) is not in mixedCase
Parameter TokenMetadata.add(address,address)._vToken (contracts/venus/
    ↪ TokenMetadata.sol#19) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #conformance-to-solidity-naming-conventions

Variable IUniswapV2Router.addLiquidity(address,address,uint256,uint256,
    ↪ uint256,uint256,address,uint256).amountADesired (contracts/
    ↪ interfaces/IUniswapV2Router.sol#12) is too similar to
    ↪ IUniswapV2Router.addLiquidity(address,address,uint256,uint256,
    ↪ uint256,uint256,address,uint256).amountBDesired (contracts/
    ↪ interfaces/IUniswapV2Router.sol#13)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #variable-names-are-too-similar

Variable IndexFactory.constructor(address,address,address,address,
    ↪ address,address,address)._baseRebalancingAddres (contracts/
    ↪ IndexFactory.sol#50) is too similar to IndexFactory.
    ↪ baseRebalancingAddress (contracts/IndexFactory.sol#19)
Variable IndexSwap.MAX_INVESTMENTAMOUNT (contracts/core/IndexSwap.sol
    ↪ #69) is too similar to IndexSwap.constructor(string,string,
    ↪ address,address,uint256,address,address,address,address,uint256,
    ↪ address)._maxInvestmentAmount (contracts/core/IndexSwap.sol#93)
Variable IUniswapV2Router02.addLiquidity(address,address,uint256,uint256
    ↪ ,uint256,uint256,address,uint256).amountADesired (contracts/
    ↪ interfaces/IUniswapV2Router02.sol#12) is too similar to
    ↪ IUniswapV2Router02.addLiquidity(address,address,uint256,uint256,
    ↪ uint256,uint256,address,uint256).amountBDesired (contracts/
    ↪ interfaces/IUniswapV2Router02.sol#13)
```

```
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #variable-names-are-too-similar

Clones.clone(address) (node_modules/@openzeppelin/contracts/proxy/Clones
    ↪ .sol#25-34) uses literals with too many digits:
 - mstore(uint256,uint256)(ptr_clone_asm_0,0
     ↪ x3d602d80600a3d3981f3363d3d373d3d3d363d730000000000000000000000000000
     ↪ ) (node_modules/@openzeppelin/contracts/proxy/Clones.sol#28)
Clones.clone(address) (node_modules/@openzeppelin/contracts/proxy/Clones
    ↪ .sol#25-34) uses literals with too many digits:
 - mstore(uint256,uint256)(ptr_clone_asm_0 + 0x28,0
     ↪ x5af43d82803e903d91602b57fd5bf3000000000000000000000000000000000000
     ↪ ) (node_modules/@openzeppelin/contracts/proxy/Clones.sol#30)
Clones.cloneDeterministic(address,bytes32) (node_modules/@openzeppelin/
    ↪ contracts/proxy/Clones.sol#43-52) uses literals with too many
    ↪ digits:
 - mstore(uint256,uint256)(ptr_cloneDeterministic_asm_0,0
     ↪ x3d602d80600a3d3981f3363d3d373d3d3d363d730000000000000000000000000000
     ↪ ) (node_modules/@openzeppelin/contracts/proxy/Clones.sol#46)
Clones.cloneDeterministic(address,bytes32) (node_modules/@openzeppelin/
    ↪ contracts/proxy/Clones.sol#43-52) uses literals with too many
    ↪ digits:
 - mstore(uint256,uint256)(ptr_cloneDeterministic_asm_0 + 0x28,0
     ↪ x5af43d82803e903d91602b57fd5bf3000000000000000000000000000000000000
     ↪ ) (node_modules/@openzeppelin/contracts/proxy/Clones.sol#48)
Clones.predictDeterministicAddress(address,bytes32,address) (
    ↪ node_modules/@openzeppelin/contracts/proxy/Clones.sol#57-72) uses
    ↪  literals with too many digits:
 - mstore(uint256,uint256)(ptr_predictDeterministicAddress_asm_0,0
     ↪ x3d602d80600a3d3981f3363d3d373d3d3d363d730000000000000000000000000000
     ↪ ) (node_modules/@openzeppelin/contracts/proxy/Clones.sol#64)
Clones.predictDeterministicAddress(address,bytes32,address) (
    ↪ node_modules/@openzeppelin/contracts/proxy/Clones.sol#57-72) uses
    ↪  literals with too many digits:
```

```
 - mstore(uint256,uint256)(ptr_predictDeterministicAddress_asm_0 + 0x28
    ↪ ,0
    ↪ x5af43d82803e903d91602b57fd5bf3ff0000000000000000000000000000000000
    ↪ ) (node_modules/@openzeppelin/contracts/proxy/Clones.sol#66)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
   ↪ #too-many-digits

BaseGuard (node_modules/@gnosis.pm/zodiac/contracts/guard/BaseGuard.sol
   ↪ #8-38) does not implement functions:
 - BaseGuard.checkAfterExecution(bytes32,bool) (node_modules/@gnosis.pm/
    ↪ zodiac/contracts/guard/BaseGuard.sol#37)
 - BaseGuard.checkTransaction(address,uint256,bytes,Enum.Operation,
    ↪ uint256,uint256,uint256,address,address,bytes,address) (
    ↪ node_modules/@gnosis.pm/zodiac/contracts/guard/BaseGuard.sol
    ↪ #23-35)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
   ↪ #unimplemented-functions

OwnableUpgradeable.__gap (node_modules/@openzeppelin/contracts-
   ↪ upgradeable/access/OwnableUpgradeable.sol#87) is never used in
   ↪ VelvetSafeModule (contracts/vault/VelvetSafeModule.sol#19-73)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
   ↪ #unused-state-variable

IndexSwap.indexPrice (contracts/core/IndexSwap.sol#42) should be
   ↪ constant
VBep20Storage.underlying (contracts/venus/VBep20Storage.sol#8) should be
   ↪  constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
   ↪ #state-variables-that-could-be-declared-constant

renounceOwnership() should be declared external:
 - OwnableUpgradeable.renounceOwnership() (node_modules/@openzeppelin/
    ↪ contracts-upgradeable/access/OwnableUpgradeable.sol#59-61)
```

```
grantRole(bytes32,address) should be declared external:
 - AccessControl.grantRole(bytes32,address) (node_modules/@openzeppelin/
    ↪ contracts/access/AccessControl.sol#142-144)
revokeRole(bytes32,address) should be declared external:
 - AccessControl.revokeRole(bytes32,address) (node_modules/@openzeppelin
    ↪ /contracts/access/AccessControl.sol#155-157)
renounceRole(bytes32,address) should be declared external:
 - AccessControl.renounceRole(bytes32,address) (node_modules/
    ↪ @openzeppelin/contracts/access/AccessControl.sol#173-177)
renounceOwnership() should be declared external:
 - Ownable.renounceOwnership() (node_modules/@openzeppelin/contracts/
    ↪ access/Ownable.sol#54-56)
transferOwnership(address) should be declared external:
 - Ownable.transferOwnership(address) (node_modules/@openzeppelin/
    ↪ contracts/access/Ownable.sol#62-65)
name() should be declared external:
 - ERC20.name() (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.
    ↪ sol#62-64)
symbol() should be declared external:
 - ERC20.symbol() (node_modules/@openzeppelin/contracts/token/ERC20/
    ↪ ERC20.sol#70-72)
decimals() should be declared external:
 - ERC20.decimals() (node_modules/@openzeppelin/contracts/token/ERC20/
    ↪ ERC20.sol#87-89)
transfer(address,uint256) should be declared external:
 - ERC20.transfer(address,uint256) (node_modules/@openzeppelin/contracts
    ↪ /token/ERC20/ERC20.sol#113-117)
approve(address,uint256) should be declared external:
 - ERC20.approve(address,uint256) (node_modules/@openzeppelin/contracts/
    ↪ token/ERC20/ERC20.sol#136-140)
transferFrom(address,address,uint256) should be declared external:
 - ERC20.transferFrom(address,address,uint256) (node_modules/
    ↪ @openzeppelin/contracts/token/ERC20/ERC20.sol#158-167)
increaseAllowance(address,uint256) should be declared external:
```

```
 - ERC20.increaseAllowance(address,uint256) (node_modules/@openzeppelin/
    ↪ contracts/token/ERC20/ERC20.sol#181-185)
decreaseAllowance(address,uint256) should be declared external:
 - ERC20.decreaseAllowance(address,uint256) (node_modules/@openzeppelin/
    ↪ contracts/token/ERC20/ERC20.sol#201-210)
burn(uint256) should be declared external:
 - ERC20Burnable.burn(uint256) (node_modules/@openzeppelin/contracts/
    ↪ token/ERC20/extensions/ERC20Burnable.sol#20-22)
burnFrom(address,uint256) should be declared external:
 - ERC20Burnable.burnFrom(address,uint256) (node_modules/@openzeppelin/
    ↪ contracts/token/ERC20/extensions/ERC20Burnable.sol#35-38)
createIndex(string,string,uint256,uint256,bool) should be declared
   ↪ external:
 - IndexFactory.createIndex(string,string,uint256,uint256,bool) (
    ↪ contracts/IndexFactory.sol#64-166)
initializeTokens(uint256,address[],uint96[]) should be declared external
   ↪ :
 - IndexFactory.initializeTokens(uint256,address[],uint96[]) (contracts/
    ↪ IndexFactory.sol#178-188)
setIndexSwapLibrary(address) should be declared external:
 - IndexFactory.setIndexSwapLibrary(address) (contracts/IndexFactory.sol
    ↪ #190-193)
setOutAsset(address) should be declared external:
 - IndexFactory.setOutAsset(address) (contracts/IndexFactory.sol
    ↪ #195-198)
setupRole(bytes32,address) should be declared external:
 - AccessController.setupRole(bytes32,address) (contracts/access/
    ↪ AccessController.sol#76-78)
_pullFromVault(IIndexSwap,address,uint256,address) should be declared
   ↪ external:
 - Adapter._pullFromVault(IIndexSwap,address,uint256,address) (contracts
    ↪ /core/Adapter.sol#76-99)
_swapETHToToken(address,uint256,address,uint256) should be declared
   ↪ external:
```

```
 - Adapter._swapETHToToken(address,uint256,address,uint256) (contracts/
    ↪ core/Adapter.sol#108-148)
_swapTokenToETH(address,uint256,address,uint256) should be declared
   ↪ external:
 - Adapter._swapTokenToETH(address,uint256,address,uint256) (contracts/
    ↪ core/Adapter.sol#158-216)
investInFund(uint256) should be declared external:
 - IndexSwap.investInFund(uint256) (contracts/core/IndexSwap.sol
    ↪ #170-226)
withdrawFund(uint256,uint256,bool) should be declared external:
 - IndexSwap.withdrawFund(uint256,uint256,bool) (contracts/core/
    ↪ IndexSwap.sol#233-324)
setPaused(bool) should be declared external:
 - IndexSwap.setPaused(bool) (contracts/core/IndexSwap.sol#374-376)
updateRecords(address[],uint96[]) should be declared external:
 - IndexSwap.updateRecords(address[],uint96[]) (contracts/core/IndexSwap
    ↪ .sol#384-399)
getTokens() should be declared external:
 - IndexSwap.getTokens() (contracts/core/IndexSwap.sol#401-403)
getRecord(address) should be declared external:
 - IndexSwap.getRecord(address) (contracts/core/IndexSwap.sol#405-407)
updateTokenList(address[]) should be declared external:
 - IndexSwap.updateTokenList(address[]) (contracts/core/IndexSwap.sol
    ↪ #409-414)
deleteRecord(address) should be declared external:
 - IndexSwap.deleteRecord(address) (contracts/core/IndexSwap.sol
    ↪ #416-418)
updateTreasury(address) should be declared external:
 - IndexSwap.updateTreasury(address) (contracts/core/IndexSwap.sol
    ↪ #420-425)
getTokenAndVaultBalance(IIndexSwap) should be declared external:
 - IndexSwapLibrary.getTokenAndVaultBalance(IIndexSwap) (contracts/core/
    ↪ IndexSwapLibrary.sol#45-105)
getTokenBalance(IIndexSwap,address,bool) should be declared external:
```

```
 - IndexSwapLibrary.getTokenBalance(IIndexSwap,address,bool) (contracts/
    ↪ core/IndexSwapLibrary.sol#111-129)
calculateSwapAmounts(IIndexSwap,uint256,uint256[],uint256) should be
   ↪ declared external:
 - IndexSwapLibrary.calculateSwapAmounts(IIndexSwap,uint256,uint256[],
    ↪ uint256) (contracts/core/IndexSwapLibrary.sol#139-154)
_getTokenPriceUSDETH(uint256) should be declared external:
 - IndexSwapLibrary._getTokenPriceUSDETH(uint256) (contracts/core/
    ↪ IndexSwapLibrary.sol#170-176)
mint(address,uint256) should be declared external:
 - ERC20Mock.mint(address,uint256) (contracts/mock/ERC20Mock.sol#16-18)
burn(address,uint256) should be declared external:
 - ERC20Mock.burn(address,uint256) (contracts/mock/ERC20Mock.sol#20-22)
transferInternal(address,address,uint256) should be declared external:
 - ERC20Mock.transferInternal(address,address,uint256) (contracts/mock/
    ↪ ERC20Mock.sol#24-30)
approveInternal(address,address,uint256) should be declared external:
 - ERC20Mock.approveInternal(address,address,uint256) (contracts/mock/
    ↪ ERC20Mock.sol#32-38)
getAggregatorInterface() should be declared external:
 - PriceOracle.getAggregatorInterface() (contracts/oracle/PriceOracle.
    ↪ sol#19)
_addFeed(address,address,AggregatorV2V3Interface) should be declared
   ↪ external:
 - PriceOracle._addFeed(address,address,AggregatorV2V3Interface) (
    ↪ contracts/oracle/PriceOracle.sol#41-52)
_updateFeed(address,address,AggregatorV2V3Interface) should be declared
   ↪ external:
 - PriceOracle._updateFeed(address,address,AggregatorV2V3Interface) (
    ↪ contracts/oracle/PriceOracle.sol#60-66)
getUsdEthPrice(uint256) should be declared external:
 - PriceOracle.getUsdEthPrice(uint256) (contracts/oracle/PriceOracle.sol
    ↪ #109-120)
getPriceTokenUSD(address,uint256) should be declared external:
```

```
 - PriceOracle.getPriceTokenUSD(address,uint256) (contracts/oracle/
    ↪ PriceOracle.sol#143-151)
setPause(bool) should be declared external:
 - Rebalancing.setPause(bool) (contracts/rebalance/Rebalancing.sol
    ↪ #84-106)
updateWeights(uint96[],uint256) should be declared external:
 - Rebalancing.updateWeights(uint96[],uint256) (contracts/rebalance/
    ↪ Rebalancing.sol#249-262)
updateTokens(address[],uint96[],uint256) should be declared external:
 - Rebalancing.updateTokens(address[],uint96[],uint256) (contracts/
    ↪ rebalance/Rebalancing.sol#291-362)
feeModule() should be declared external:
 - Rebalancing.feeModule() (contracts/rebalance/Rebalancing.sol#365-440)
updateTreasury(address) should be declared external:
 - Rebalancing.updateTreasury(address) (contracts/rebalance/Rebalancing.
    ↪ sol#442-444)
transferModuleOwnership(address) should be declared external:
 - Vault.transferModuleOwnership(address) (contracts/vault/Vault.sol
    ↪ #19-25)
executeTransactionETH(address,uint256) should be declared external:
 - Vault.executeTransactionETH(address,uint256) (contracts/vault/Vault.
    ↪ sol#27-34)
executeTransactionOther(address,uint256,address) should be declared
   ↪ external:
 - Vault.executeTransactionOther(address,uint256,address) (contracts/
    ↪ vault/Vault.sol#36-42)
transferModuleOwnership(address) should be declared external:
 - VelvetSafeModule.transferModuleOwnership(address) (contracts/vault/
    ↪ VelvetSafeModule.sol#33-39)
executeTransactionETH(address,uint256) should be declared external:
 - VelvetSafeModule.executeTransactionETH(address,uint256) (contracts/
    ↪ vault/VelvetSafeModule.sol#52-58)
executeTransactionOther(address,uint256,address) should be declared
   ↪ external:
```

```
 - VelvetSafeModule.executeTransactionOther(address,uint256,address) (
    ↪ contracts/vault/VelvetSafeModule.sol#60-72)
add(address,address) should be declared external:
 - TokenMetadata.add(address,address) (contracts/venus/TokenMetadata.sol
    ↪ #19-29)
addBNB() should be declared external:
 - TokenMetadata.addBNB() (contracts/venus/TokenMetadata.sol#31-38)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
   ↪ #public-function-that-could-be-declared-external
. analyzed (60 contracts with 77 detectors), 346 result(s) found
```

Conclusion:

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

7 Conclusion

In this audit, we examined the design and implementation of Velvet Capital contract and discovered several issues of varying severity. Velvet Capital team addressed 20 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 Velvet Capital 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