Bracket.fi - BracketX

Smart Contract Security Audit

Prepared by: **Halborn**

Date of Engagement: **September 12th, 2022 - September 30th, 2022**

Visit: **Halborn.com**

DOCUMENT REVISION HISTORY

VERSION	MODIFICATION	DATE	AUTHOR
0.1	Document Creation	09/26/2022	Manuel García
0.2	Document Update	09/27/2022	Manuel García
0.3	Document Update	09/30/2022	Manuel García
0.4	Draft Review	09/30/2022	Kubilay Onur Gungor
0.5	Draft Review	09/30/2022	Gabi Urrutia
1.0	Remediation Plan	10/06/2022	Manuel García
1.1	Remediation Plan Review	10/11/2022	Kubilay Onur Gungor
1.2	Remediation Plan Review	10/11/2022	Gabi Urrutia

CONTACTS

CONTACT	COMPANY	EMAIL
Rob Behnke	Halborn	Rob.Behnke@halborn.com
Steven Walbroehl	Halborn	Steven.Walbroehl@halborn.com

Gabi Urrutia	Halborn	Gabi.Urrutia@halborn.com
Kubilay Onur Gungor	Halborn	Kubilay.Gungor@halborn.com
Manuel García	Halborn	Manuel.Diaz@halborn.com

EXECUTIVE OVERVIEW

1.1 INTRODUCTION

Bracket.fi engaged Halborn to conduct a security audit on their smart contracts beginning on September 12th, 2022 and ending on September 30th, 2022. The security assessment was scoped to the smart contracts provided in the GitLab repository bracketlabs/Bracketx Halborn.

1.2 AUDIT SUMMARY

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

The purpose of this audit is to:

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

In summary, Halborn identified some security risks that were addressed by the Bracket.fi team.

1.3 TEST APPROACH & METHODOLOGY

Halborn performed a combination of manual and automated security testing to balance efficiency, timeliness, practicality, and accuracy in regard to the scope of this audit. While manual testing is recommended to uncover flaws in logic, process, and implementation; automated testing techniques help enhance coverage of the code and can quickly identify items that do not follow the security best practices. The following phases and associated tools were used during the audit:

- Research into architecture and purpose
- Smart contract manual code review and walkthrough
- Graphing out functionality and contract logic/connectivity/functions (solgraph)
- Manual assessment of use and safety for the critical Solidity variables and functions in scope to identify any arithmetic related vulnerability classes
- Manual testing by custom scripts
- Scanning of solidity files for vulnerabilities, security hot-spots or bugs. (MythX)
- Static Analysis of security for scoped contract, and imported functions. (Slither)
- Testnet deployment (Brownie, Remix IDE)

RISK METHODOLOGY:

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

RISK SCALE - LIKELIHOOD

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

RISK SCALE - IMPACT

5 - May cause devastating and unrecoverable impact or loss.
4 - May cause a significant level of impact or loss.

3 - May cause a partial impact or loss to many.
2 - May cause temporary impact or loss.
1 - May cause minimal or un-noticeable impact.

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

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

1.4 SCOPE

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

- Bracketx.sol
- Config.sol
- IPricing.sol
- Offersx.sol
- PricingSequencer.sol
- StructLibx.sol
- iBNFT.sol

Initial Commit ID:
- 64c7ef710b03aee4125232c87e194f0e717ba0ab

Fixed Commit IDs:
- 72286e3465a5ac77b5e9142cfef70e7a07ee1faf
- 803574d1c2695a50a8fb38991d556a38a1fa23cb
- f8e46f1a94933f6418d2e67fb75a9251a0f38043
- 5d7a571ad84c5ff733556eaf4e475d51d5a99165

2. ASSESSMENT SUMMARY & FINDINGS OVERVIEW

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL
0	0	0	3	4

LIKELIHOOD

IMPACT

(HAL-02) (HAL-03)	(HAL-01)			
(HAL-04) (HAL-05) (HAL-06) (HAL-07)				

EXECUTIVE OVERVIEW

SECURITY ANALYSIS	RISK LEVEL	REMEDIATION DATE
HAL01 - CLAIMED POLICIES CAN BE TRANSFERED	Low	SOLVED - 09/30/2022
HAL02 - LACK OF DISABLEINITIALIZERS CALL TO PREVENT UNINITIALIZED CONTRACTS	Low	SOLVED - 09/30/2022
HAL03 - LACK OF PRICE FEED DECIMALS CHECK	Low	SOLVED - 09/30/2022
HAL04 - > 0 CONSUMES MORE GAS THAN != 0	Informational	SOLVED - 10/02/2022
HAL05 - POSTFIX OPERATORS CONSUME MORE GAS THAN PREFIX OPERATORS	Informational	SOLVED - 10/02/2022
HAL06 - INCREMENTS CAN BE UNCHECKED IN LOOPS	Informational	SOLVED - 10/02/2022
HAL07 - MISSING ZERO ADDRESS CHECK	Informational	SOLVED - 10/02/2022

FINDINGS & TECH DETAILS

3.1 (HAL-01) CLAIMED POLICIES CAN BE TRANSFERED - LOW

Description:

The transferFrom() function in the iBNFT.sol contract overrides the OpenZeppelin's ERC721 function to implement checks that prevent a policy NFT from being transferred if the token ID is 0 or above the current ID and prevent the ownership transfer of policies that have been already claimed:

Listing 1: iBNFT.sol (Line 83)

```
78 // Get a positive token price from a chainlink oracle
79 function transferFrom(
80     address from,
81     address to,
82     uint256 tokenId
83 ) whenNotPaused() nonReentrant() public override(ERC721Upgradeable
↳ , IERC721Upgradeable ) {
84     require( tokenId >= 1 && tokenId <= _tokenIds.current() && !
↳ Policies[tokenId].claimed, "CLAIMED");
85     super.transferFrom(from, to, tokenId);
86 }
```

However, the safeTransferFrom() function is not overridden. This allows a user to easily bypass this requirement by using the safeTransferFrom() function instead of transferFrom().

Proof of Concept:

- User buys a policy and receives a NFT with token ID 1.
- After policy expires, or the option is In-The-Money buyer claims the policy and funds are distributed accordingly.
- User tries to transfer the policy to another user using the transferFrom() function iBNFTaddr.transferFrom(buyer1, accounts[3], 1, {'from': buyer1})

- Because policy is claimed transaction reverts.

```
Transaction sent: 0xe63b3df1d9de809adbe54adecdc15049de14f86532549a8e65a1d6035146d6f4
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 23
  iBNFT.transferFrom confirmed (CLAIMED)   Block: 143   Gas used: 31011 (0.26%)
```

- User calls the safeTransferFrom() function instead. Effectively transferring the NFT ownership and bypassing the restriction. iBNFTaddr.safeTransferFrom(buyer1, accounts[3], 1, {'from': buyer1 })

```
Transaction sent: 0x1db91edc930962c2321aa7bd452a3d55ad3c9db900acaa79c9fbc49243c4705f
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 27
  iBNFT.safeTransferFrom confirmed   Block: 164   Gas used: 65294 (0.54%)
```

Risk Level:

Likelihood - 2
Impact - 3

Recommendation:

It is recommended to override the safeTransferFrom() function and add the same check on the iBNFT contract:

Listing 2: iBNFT.sol

```
1 function safeTransferFrom(
2     address from,
3     address to,
4     uint256 tokenId
5 ) whenNotPaused() nonReentrant() public override(ERC721Upgradeable
↳ , IERC721Upgradeable ) {
6     require( tokenId >= 1 && tokenId <= _tokenIds.current() && !
↳ Policies[tokenId].claimed, "CLAIMED");
7     super.safeTransferFrom(from, to, tokenId);
8 }
```

Remediation Plan:

SOLVED: The Bracket.fi team fixed the issue. Both safeTransferFrom functions, one with 3 and the other with 4 parameters, were overridden

in the iBNFT contract. Moreover, the requirements were transferred to the 4 safeTransferFrom arguments, while the remaining transfer functions call it internally.

3.2 (HAL-02) LACK OF DISABLEINITIALIZERS CALL TO PREVENT UNINITIALIZED CONTRACTS - LOW

Description:

Multiple contracts are using the Initializable module from OpenZeppelin. In order to prevent leaving an implementation contract uninitialized OpenZeppelin's documentation recommends adding the _disableInitializers function in the constructor to lock the contracts automatically when they are deployed:

Listing 3

```
/**
 * @dev Locks the contract, preventing any future reinitialization. This cannot be part of an initializer call.
 * Calling this in the constructor of a contract will prevent that contract from being initialized or reinitialized
 * to any version. It is recommended to use this to lock implementation contracts that are designed to be called
 * through proxies.
 */
function _disableInitializers() internal virtual {
    require(!_initializing, "Initializable: contract is initializing");
    if (_initialized < type(uint8).max) {
        _initialized = type(uint8).max;
        emit Initialized(type(uint8).max);
    }
}
```

Risk Level:

Likelihood - 1
Impact - 3

Recommendation:

Consider calling the _disableInitializers function in the contract constructor:

Listing 4

```
/// @custom:oz-upgrades-unsafe-allow constructor
constructor() {
    _disableInitializers();
}
```

Remediation Plan:

SOLVED: The Bracket.fi team fixed the issue. Added the _disableInitializers() to the following contracts: Bracketxl, Config, Offersx, PricingSequencer, iBNFT.

3.3 (HAL-03) LACK OF PRICE FEED DECIMALS CHECK - LOW

Description:

The PricingSequencer contract contains the getLatestPrice() function to return the latest price from a given Chainlink price feed address. This is intended to be used with USDC pairs, which will return an 8 decimals price which is later on converted to 18 decimals by multiplying it by 1e10. However, a funder can create an offer sending an asset that would return an 18-decimals price; this would lead to the function returning a 28-decimals price.

Listing 5: PricingSequencer.sol (Line 542)

```
526 function getLatestPrice(address asset) override public view
 ↳ returns (uint) {
527     // TODO: Add the Sequencer offline check when moving to
 ↳ production
528     // this is not supported on the testnet
529
530     if (checkSequencerState()) {
531         // If the sequencer is down, do not perform any critical
 ↳ operations
532         revert("L2 sequencer down: Chainlink feeds are not being
 ↳ updated");
533     }
534
535     //        uint80 roundID,
536     //            int price,
537     //            uint startedAt,
538     //            uint timeStamp,
539     //            uint80 answeredInRound
540     AggregatorV2V3Interface priceFeed = AggregatorV2V3Interface(
 ↳ asset);
541     (, int price, ,,) = priceFeed.latestRoundData();
542     return uint256(price).mul(1e10);
543 }
```

Risk Level:

Likelihood - 1
Impact - 3

Recommendation:

Although the probability of this happening is reduced as the Bracket.fi team has confirmed that they are filtering the offers by asset on the front-end; Halborn recommends whitelisting the assets or checking the decimals of the provided price feed address. This could be done in the setOfferX() function of the Offersx contract, so no funder can set an offer that returns a price other than 8 decimals price.

Remediation Plan:

SOLVED: The Bracket.fi team fixed the issue by adding code to the setOfferX function in the Offersx contract that checks that oracle uses 8 decimals before setting an offer. Although the code is commented out like oracles, it cannot be tested in local environments. Comments should be removed before deploying to production.

3.4 (HAL-04) > 0 CONSUMES MORE GAS THAN != 0 - INFORMATIONAL

Description:

The use of > consumes more gas than !=. There are some cases where both can be used indistinctly, such as in unsigned integers where numbers can't be negative, and as such, there is only a need to check that a number is not 0.

Code Location:

Bracketx.sol
- Line 244: `assert(v.id > 0);`
- Line 376: `if (v.addAvailUSDC > 0){`
- Line 419: `require((amountETH > 0 || amountUSDC > 0), "BAL");`
- Line 420: `if (amountETH > 0){`
- Line 424: `if (amountUSDC > 0){`

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

Use != instead of > in cases where both can be used.

Remediation Plan:

SOLVED: The Bracket.fi team fixed the issue by replacing > with != in the specified code.

3.5 (HAL-05) POSTFIX OPERATORS CONSUME MORE GAS THAN PREFIX OPERATORS - INFORMATIONAL

Description:

The use of postfix operators `i++` consume more gas than prefix operators `++i`.

Code Location:

Bracketx.sol
- Line 397: `for (uint tid = autoClaimPtr; tid <= maxId; tid++ ){`
- Line 405: `cnt++;`

Offersx.sol
- Line 71: `for (uint8 i; i < 12; i++){`

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

It is recommended to use prefix operators rather than postfix.

Remediation Plan:

SOLVED: The Bracket.fi team fixed the issue by replacing the postfix with prefix operators.

3.6 (HAL-06) INCREMENTS CAN BE UNCHECKED IN LOOPS - INFORMATIONAL

Description:

Most of the solidity for loops use an uint256 variable counter that increments by 1 and starts at 0. These increments don't need to be checked for over/underflow because the variable will never reach the max capacity of uint256 as it would run out of gas long before that happens.

Code Location:

Bracketx.sol
- Line 397: `for (uint tid = autoClaimPtr; tid <= maxId; tid++ ){`
- Line 405: `cnt++;`

Offersx.sol
- Line 71: `for (uint8 i; i < 12; i++){`

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

It is recommended to uncheck the increments in for loops to save gas. For example, instead of:

Listing 6: BracketX.sol

```
397 for (uint tid = autoClaimPtr; tid <= maxId; tid++ ) {
398     // Code to be run
399 }
```

It could be used to save gas:

Listing 7: BracketX.sol

```
397 for (uint tid = autoClaimPtr; tid <= maxId;) {
398     unchecked { ++tid; }
399 }
```

Remediation Plan:

SOLVED: The Bracket.fi team fixed the issue by unchecking increments in for loops.

3.7 (HAL-07) MISSING ZERO ADDRESS CHECK - INFORMATIONAL

Description:

It has been detected that constructors or initializing functions of many smart contracts are missing address validation. For example:

Listing 8: Bracketx.sol (Lines 96-100)

```
85 function initialize(
86     address _ibnft,
87     address _usdc,
88     address _pricing,
89     address _offers,
90     address _config
91 ) external initializer {
92     __Ownable_init();
93     __ReentrancyGuard_init();
94     __Pausable_init();
95
96     ibnft = _ibnft;
97     usdc = _usdc;
98     pricing = _pricing;
99     offers = _offers;
100    config = _config;
101    autoClaimPtr = 1; // first nft is 1 not 0.
102 }
```

Every input address should be checked not to be zero, especially the ones that could lead to rendering the contract unusable, lock tokens, etc. This is considered a best practice.

Code Location:

Bracketx.sol
- Line 85-102: initialize()

iBNFT.sol
- Line 40-42: setBrkt()

Offersx.sol
- Line 39-46: initialize()

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

It is recommended to validate that each address inputs in the constructor and other critical functions are non-zero.

Remediation Plan:

SOLVED: The Bracket.fi team fixed the issue by implementing non-zero address requirements in the initialized function of contracts.

MANUAL TESTING

MANUAL TESTING

Halborn performed several manual tests in the following contracts:

- BracketX.sol
- Config.sol
- iBNFT.sol
- PricingSequencer.sol
- Offersx.sol
- StructLibx.sol

The manual tests were focused on testing the main functions of these contracts:

- addAvailable()
- reduceAvailable()
- buyPolicy()
- claim()
- autoClaim()
- checkVer()
- addVer()
- transferFrom()
- mintNFTinsured()
- getOffer()
- setOfferx()
- getLatestPrice()
- checkSequencerState()
- multFactor()

No issues were found during the manual tests.

AUTOMATED TESTING

5.1 STATIC ANALYSIS REPORT

Description:

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

Slither results:

BracketX.sol

```
Brackets.withdrawU (jobs/bracket/brackets/brownie/contracts/Brackets.sol#416-429) ignores return value by IERC20(usdc).transfer(msg.sender,amountUSDC) (jobs/bracket/brackets/brownie/contracts/Brackets.sol#426)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-transfer

MathUpgradeable.mulDiv(uint256,uint256,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#55-135) performs a multiplication on the result of a division:
        -denominator = denominator / twos (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#102)
        -inverse = (3 * denominator) ^ 2 (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#117)
MathUpgradeable.mulDiv(uint256,uint256,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#55-135) performs a multiplication on the result of a division:
        -denominator = denominator / twos (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#102)
        -inverse *= 2 - denominator * inverse (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#121)
MathUpgradeable.mulDiv(uint256,uint256,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#55-135) performs a multiplication on the result of a division:
        -denominator = denominator / twos (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#102)
        -inverse *= 2 - denominator * inverse (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#122)
MathUpgradeable.mulDiv(uint256,uint256,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#55-135) performs a multiplication on the result of a division:
        -denominator = denominator / twos (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#102)
        -inverse *= 2 - denominator * inverse (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#123)
MathUpgradeable.mulDiv(uint256,uint256,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#55-135) performs a multiplication on the result of a division:
        -denominator = denominator / twos (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#102)
        -inverse *= 2 - denominator * inverse (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#124)
MathUpgradeable.mulDiv(uint256,uint256,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#55-135) performs a multiplication on the result of a division:
        -denominator = denominator / twos (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#102)
        -inverse *= 2 - denominator * inverse (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#125)
MathUpgradeable.mulDiv(uint256,uint256,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#55-135) performs a multiplication on the result of a division:
        -denominator = denominator / twos (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#102)
        -inverse *= 2 - denominator * inverse (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#126)
MathUpgradeable.mulDiv(uint256,uint256,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#55-135) performs a multiplication on the result of a division:
        -prod0 = prod0 / twos (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#105)
        -result = prod0 * inverse (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#132)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply

Brackets.claim(uint256) (jobs/bracket/brackets/brownie/contracts/Brackets.sol#312-392) uses a dangerous strict equality:
        - require(bool,string)(msg.sender == v.owner,NOT OWNER) (jobs/bracket/brackets/brownie/contracts/Brackets.sol#322)
Brackets.claim(uint256) (jobs/bracket/brackets/brownie/contracts/Brackets.sol#312-392) uses a dangerous strict equality:
        - v.p.isLong == true (jobs/bracket/brackets/brownie/contracts/Brackets.sol#328)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities

Reentrancy in Brackets.autoClaim(uint16) (jobs/bracket/brackets/brownie/contracts/Brackets.sol#394-413):
        External calls:
        - claim(tid) (jobs/bracket/brackets/brownie/contracts/Brackets.sol#404)
                - IBNFT(ibnft).setClaimed(_policyNum) (jobs/bracket/brackets/brownie/contracts/Brackets.sol#354)
        State variables written after the call(s):
        - autoClaimPtr += 1 (jobs/bracket/brackets/brownie/contracts/Brackets.sol#409)
Reentrancy in Brackets.buyPolicy(address,address,uint8,uint256,uint256) (jobs/bracket/brackets/brownie/contracts/Brackets.sol#172-294):
        External calls:
        - (success2,data2) = address(ibnft).call(abi.encodeWithSignature(mintNFTInsured(address),msg.sender)) (jobs/bracket/brackets/brownie/contracts/Brackets.sol#239-241)
        State variables written after the call(s):
        - FundersUtil_asset[_funder] = v.availUSD.div(1e12) (jobs/bracket/brackets/brownie/contracts/Brackets.sol#266)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1

Brackets.claim(uint256).v (jobs/bracket/brackets/brownie/contracts/Brackets.sol#313) is a local variable never initialized
Brackets.buyPolicy(address,address,uint8,uint256,uint256).v (jobs/bracket/brackets/brownie/contracts/Brackets.sol#179) is a local variable never initialized
Offerx.setOfferX(address,uint256[12],uint16,i) (jobs/bracket/brackets/brownie/contracts/Offerx.sol#71) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables

ERC721Upgradeable._checkOnERC721Received(address,address,uint256,bytes) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#399-421) ignores return value by IERC721ReceiverUpgradeable(to).onERC721Received(_msgSender(),from,tok
enId,data) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#406-417)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
:IBNFT.setBrkt(address) (jobs/bracket/brackets/brownie/contracts/IBNFT.sol#40-42) should emit an event for:
        - brkt = _brkt (jobs/bracket/brackets/brownie/contracts/IBNFT.sol#41)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-access-control

Brackets.initialize(address,address,address,address,address)._ibnft (jobs/bracket/brackets/brownie/contracts/Brackets.sol#86) lacks a zero-check on :
        - ibnft = _ibnft (jobs/bracket/brackets/brownie/contracts/Brackets.sol#90)
Brackets.initialize(address,address,address,address,address)._usdc (jobs/bracket/brackets/brownie/contracts/Brackets.sol#87) lacks a zero-check on :
        - usdc = _usdc (jobs/bracket/brackets/brownie/contracts/Brackets.sol#91)
Brackets.initialize(address,address,address,address,address)._pricing (jobs/bracket/brackets/brownie/contracts/Brackets.sol#88) lacks a zero-check on :
        - pricing = _pricing (jobs/bracket/brackets/brownie/contracts/Brackets.sol#98)
Brackets.initialize(address,address,address,address,address)._offers (jobs/bracket/brackets/brownie/contracts/Brackets.sol#89) lacks a zero-check on :
        - offers = _offers (jobs/bracket/brackets/brownie/contracts/Brackets.sol#99)
Brackets.initialize(address,address,address,address,address)._config (jobs/bracket/brackets/brownie/contracts/Brackets.sol#90) lacks a zero-check on :
        - config = _config (jobs/bracket/brackets/brownie/contracts/Brackets.sol#100)
Offerx.initialize(address,address)._pricing (jobs/bracket/brackets/brownie/contracts/Offerx.sol#39) lacks a zero-check on :
        - pricing = _pricing (jobs/bracket/brackets/brownie/contracts/Offerx.sol#44)
Offerx.initialize(address,address)._config (jobs/bracket/brackets/brownie/contracts/Offerx.sol#39) lacks a zero-check on :
        - config = _config (jobs/bracket/brackets/brownie/contracts/Offerx.sol#45)
:IBNFT.setBrkt(address)._brkt (jobs/bracket/brackets/brownie/contracts/IBNFT.sol#40) lacks a zero-check on :
        - brkt = _brkt (jobs/bracket/brackets/brownie/contracts/IBNFT.sol#41)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

Brackets.autoClaim(uint16) (jobs/bracket/brackets/brownie/contracts/Brackets.sol#394-413) has external calls inside a loop: p = IBNFT(ibnft).getPolicy(tid) (jobs/bracket/brackets/brownie/contracts/Brackets.sol#403)
Brackets.claim(uint256) (jobs/bracket/brackets/brownie/contracts/Brackets.sol#312-392) has external calls inside a loop: v.p = IBNFT(ibnft).getPolicy(_policyNum) (jobs/bracket/brackets/brownie/contracts/Brackets.sol#314)
Brackets.claim(uint256) (jobs/bracket/brackets/brownie/contracts/Brackets.sol#312-392) has external calls inside a loop: v.owner = IBNFT(ibnft).ownerOf(v.p.id) (jobs/bracket/brackets/brownie/contracts/Brackets.sol#317)
Brackets.claim(uint256) (jobs/bracket/brackets/brownie/contracts/Brackets.sol#312-392) has external calls inside a loop: v.claimPrice = IPricing(pricing).getClaimPrice(v.p.asset) (jobs/bracket/brackets/brownie/contracts/Brackets.sol#325)
Brackets.claim(uint256) (jobs/bracket/brackets/brownie/contracts/Brackets.sol#312-392) has external calls inside a loop: IBNFT(ibnft).setClaimed(_policyNum) (jobs/bracket/brackets/brownie/contracts/Brackets.sol#354)
Brackets.claim(uint256) (jobs/bracket/brackets/brownie/contracts/Brackets.sol#312-392) has external calls inside a loop: v.commishUSDC = StructLibx.multFactor(v,preclaimAntiUSD,Config(config).CLAIM_COMMISSION()).div(1e12) (jobs/bracket/brackets/brownie/contracts/Brackets.sol#360-370)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#calls-inside-a-loop

Variable 'ERC721Upgradeable._checkOnERC721Received(address,address,uint256,bytes).retval (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#406)' in ERC721Upgradeable._checkOnERC721Received(address,address,uint256,bytes) (.br
ownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#399-421) potentially used before declaration: retval == IERC721ReceiverUpgradeable.onERC721Received.selector (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contract
s-upgradeable/token/ERC721/ERC721Upgradeable.sol#407)
Variable 'ERC721Upgradeable._checkOnERC721Received(address,address,uint256,bytes).reason (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#408)' in ERC721Upgradeable._checkOnERC721Received(address,address,uint256,bytes) (.br
ownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#399-421) potentially used before declaration: reason.length == 0 (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/token/ERC721/ERC721Upgradeable
.sol#409)
Variable 'ERC721Upgradeable._checkOnERC721Received(address,address,uint256,bytes).reason (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#408)' in ERC721Upgradeable._checkOnERC721Received(address,address,uint256,bytes) (.br
ownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#399-421) potentially used before declaration: revert(uint256,uint256)(32 + reason,mload(uint256)(reason)) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-u
pgradeable/token/ERC721/ERC721Upgradeable.sol#414)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#pre-declaration-usage-of-local-variables
```

```
Reentrancy in Brackets.addAvailable(address,uint256) (jobs/bracket/brackets/brownie/contracts/Brackets.sol#125-136):
        External calls:
        - require(bool,string)(IERC20(usdc).transferFrom(msg.sender,address(this),_usdcValue),T1) (jobs/bracket/brackets/brownie/contracts/Brackets.sol#128-131)
        State variables written after the call(s):
        - FunderAvil[asset][msg.sender] = priorAmt.add(_usdcValue) (jobs/bracket/brackets/brownie/contracts/Brackets.sol#134)
Reentrancy in Brackets.buyPolicy(address,address,uint8,uint256,uint256) (jobs/bracket/brackets/brownie/contracts/Brackets.sol#172-294):
        External calls:
        - (success2,data2) = address(ibnftl).call(abi.encodeWithSignature(mintNFTinsured(address),msg.sender)) (jobs/bracket/brackets/brownie/contracts/Brackets.sol#230-241)
        State variables written after the call(s):
        - pendingETHwithdrawals[owner()] = pendingETHwithdrawals[owner()].add(v.commishETH) (jobs/bracket/brackets/brownie/contracts/Brackets.sol#260-262)
        - pendingETHwithdrawals[_funder] = pendingETHwithdrawals[_funder].add(v.funderETHpay) (jobs/bracket/brackets/brownie/contracts/Brackets.sol#263-265)
Reentrancy in Brackets.claim(uint256) (jobs/bracket/brackets/brownie/contracts/Brackets.sol#312-392):
        External calls:
        - IBNFT(ibnftl).setClaimed(_policyNum) (jobs/bracket/brackets/brownie/contracts/Brackets.sol#354)
        State variables written after the call(s):
        - FunderAvil[o.p.asset][o.p.funder] = FunderAvil[v.p.asset][e.p.funder].add(v.addAvailUSDC) (jobs/bracket/brackets/brownie/contracts/Brackets.sol#378)
        - pendingUSDCwithdrawals[owner()] = pendingUSDCwithdrawals[owner()].add(v.commishUSDC) (jobs/bracket/brackets/brownie/contracts/Brackets.sol#371)
        - pendingUSDCwithdrawals[v.owner] = pendingUSDCwithdrawals[v.owner].add(v.claimAmtUSDC) (jobs/bracket/brackets/brownie/contracts/Brackets.sol#374)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2

Reentrancy in Brackets.addAvailable(address,uint256) (jobs/bracket/brackets/brownie/contracts/Brackets.sol#125-136):
        External calls:
        - require(bool,string)(IERC20(usdc).transferFrom(msg.sender,address(this),_usdcValue),T1) (jobs/bracket/brackets/brownie/contracts/Brackets.sol#128-131)
        Event emitted after the call(s):
        - AvailableEv(asset,msg.sender,priorAmt,FunderAvil[asset][msg.sender]) (jobs/bracket/brackets/brownie/contracts/Brackets.sol#135)
Reentrancy in Brackets.buyPolicy(address,address,uint8,uint256,uint256) (jobs/bracket/brackets/brownie/contracts/Brackets.sol#172-294):
        External calls:
        - (success2,data2) = address(ibnftl).call(abi.encodeWithSignature(mintNFTinsured(address),msg.sender)) (jobs/bracket/brackets/brownie/contracts/Brackets.sol#230-241)
        - IBNFT(ibnftl).setPolicy(v.id,p) (jobs/bracket/brackets/brownie/contracts/Brackets.sol#269)
        Event emitted after the call(s):
        - PolicyvBuyEv1(v.id,msg.sender,_funder,_asset,_offer1d,v.farPriceUSD,v.nearPriceUSD,msg.value,v.premiumUSD,v.funderETHpay) (jobs/bracket/brackets/brownie/contracts/Brackets.sol#271-281)
        - PolicyvBuyEv2(v.id,msg.sender,_funder,_asset,_offer1d,block.timestamp,v.timeExpired,v.maxClaimUSD,v.commishETH,v.priceCoin) (jobs/bracket/brackets/brownie/contracts/Brackets.sol#283-293)
Reentrancy in Brackets.claim(uint256) (jobs/bracket/brackets/brownie/contracts/Brackets.sol#312-392):
        External calls:
        - IBNFT(ibnftl).setClaimed(_policyNum) (jobs/bracket/brackets/brownie/contracts/Brackets.sol#354)
        Event emitted after the call(s):
        - ClaimEv(v.p.funder,_policyNum,v.p.asset,v.owner,v.claimTime,v.claimPrice,v.claimAmtUSDC,v.commishUSDC,v.addAvailUSDC) (jobs/bracket/brackets/brownie/contracts/Brackets.sol#381-391)
Reentrancy in Brackets.withdraw() (jobs/bracket/brackets/brownie/contracts/Brackets.sol#416-429):
        External calls:
        - IERC20(usdc).transfer(msg.sender,amountUSDC) (jobs/bracket/brackets/brownie/contracts/Brackets.sol#426)
        External calls sending eth:
        - address(msg.sender).transfer(amountETH) (jobs/bracket/brackets/brownie/contracts/Brackets.sol#422)
        Event emitted after the call(s):
        - WithdrawEv(msg.sender,amountETH,amountUSDC) (jobs/bracket/brackets/brownie/contracts/Brackets.sol#428)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
Brackets.buyPolicy(address,address,uint8,uint256,uint256) (jobs/bracket/brackets/brownie/contracts/Brackets.sol#172-294) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(v.p.nearbrkt[_offer1d] >= 1e16,NO OFFER) (jobs/bracket/brackets/brownie/contracts/Brackets.sol#187)
        - require(bool,string)(v.priceCoin <= _maxAssetPrice,SLIP1) (jobs/bracket/brackets/brownie/contracts/Brackets.sol#191)
        - require(bool,string)(v.priceCoin >= _minAssetPrice,SLIP2) (jobs/bracket/brackets/brownie/contracts/Brackets.sol#192)
        - require(bool,string)(v.availUSD >= v.maxClaimUSD,AVAIL) (jobs/bracket/brackets/brownie/contracts/Brackets.sol#228)
        - assert(bool)(v.id > 0) (jobs/bracket/brackets/brownie/contracts/Brackets.sol#244)
Brackets.claim(uint256) (jobs/bracket/brackets/brownie/contracts/Brackets.sol#312-392) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(v.p.claimed == false,CLAIMED) (jobs/bracket/brackets/brownie/contracts/Brackets.sol#315)
        - v.claimTime != v.p.timeExpired (jobs/bracket/brackets/brownie/contracts/Brackets.sol#321)
        - require(bool,string)(msg.sender == v.owner,NOT OWNER) (jobs/bracket/brackets/brownie/contracts/Brackets.sol#322)
        - v.p.isLong == true (jobs/bracket/brackets/brownie/contracts/Brackets.sol#328)
        - v.zeroClaim = v.claimPrice <= v.p.nearPriceUSD (jobs/bracket/brackets/brownie/contracts/Brackets.sol#329)
        - v.zeroClaim = v.claimPrice >= v.p.nearPriceUSD (jobs/bracket/brackets/brownie/contracts/Brackets.sol#337)
        - v.claimTime != v.p.timeExpired (jobs/bracket/brackets/brownie/contracts/Brackets.sol#349)
        - require(bool,string)(! v.zeroClaim,DOTHONEY) (jobs/bracket/brackets/brownie/contracts/Brackets.sol#350)
        - v.addAvailUSDC > 0 (jobs/bracket/brackets/brownie/contracts/Brackets.sol#376)
Brackets.autoClaim(uint16) (jobs/bracket/brackets/brownie/contracts/Brackets.sol#394-413) uses timestamp for comparisons
        Dangerous comparisons:
        - block.timestamp >= p.timeExpired && ! p.claimed (jobs/bracket/brackets/brownie/contracts/Brackets.sol#402)
        - block.timestamp >= p.timeExpired || p.claimed (jobs/bracket/brackets/brownie/contracts/Brackets.sol#407)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp

ERC721Upgradeable._checkOnERC721Received(address,address,uint256,bytes) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#399-421) uses assembly
        - INLINE ASM (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#413-415)
AddressUpgradeable.verifyCallResult(bool,bytes,string) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/AddressUpgradeable.sol#174-194) uses assembly
        - INLINE ASM (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/AddressUpgradeable.sol#186-189)
MathUpgradeable.mulDiv(uint256,uint256,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#55-135) uses assembly
        - INLINE ASM (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#66-70)
        - INLINE ASM (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#86-93)
        - INLINE ASM (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#100-109)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

Brackets.buyPolicy(address,address,uint8,uint256,uint256) (jobs/bracket/brackets/brownie/contracts/Brackets.sol#172-294) compares to a boolean constant:
        -v.isLong == true (jobs/bracket/brackets/brownie/contracts/Brackets.sol#211)
Brackets.claim(uint256) (jobs/bracket/brackets/brownie/contracts/Brackets.sol#312-392) compares to a boolean constant:
        -require(bool,string)(v.p.claimed == false,CLAIMED) (jobs/bracket/brackets/brownie/contracts/Brackets.sol#315)
Brackets.claim(uint256) (jobs/bracket/brackets/brownie/contracts/Brackets.sol#312-392) compares to a boolean constant:
        -v.p.isLong == true (jobs/bracket/brackets/brownie/contracts/Brackets.sol#328)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#boolean-equality

Different versions of Solidity are used:
        - Version used: ['^0.8.0', '^0.8.1', '^0.8.2', '^0.8.4']
        - ^0.8.0 (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/access/OwnableUpgradeable.sol#4)
        - ^0.8.2 (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/proxy/utils/Initializable.sol#4)
        - ^0.8.0 (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/security/PausableUpgradeable.sol#4)
        - ^0.8.0 (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/security/ReentrancyGuardUpgradeable.sol#4)
        - ^0.8.0 (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#4)
        - ^0.8.0 (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/token/ERC721/IERC721ReceiverUpgradeable.sol#4)
        - ^0.8.0 (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/token/ERC721/IERC721Upgradeable.sol#4)
        - ^0.8.0 (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/token/ERC721/extensions/ERC721EnumerableUpgradeable.sol#4)
        - ^0.8.0 (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/token/ERC721/extensions/IERC721EnumerableUpgradeable.sol#4)
        - ^0.8.0 (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/token/ERC721/extensions/IERC721MetadataUpgradeable.sol#4)
        - ^0.8.1 (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/AddressUpgradeable.sol#4)
        - ^0.8.0 (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/ContextUpgradeable.sol#4)
        - ^0.8.0 (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/CountersUpgradeable.sol#4)
        - ^0.8.0 (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/StringsUpgradeable.sol#4)
        - ^0.8.0 (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/introspection/ERC165Upgradeable.sol#4)
        - ^0.8.0 (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/introspection/IERC165Upgradeable.sol#4)
        - ^0.8.0 (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#4)
        - ^0.8.0 (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/SafeMathUpgradeable.sol#4)
        - ^0.8.0 (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts/token/ERC20/IERC20.sol#4)
        - ^0.8.4 (jobs/bracket/brackets/brownie/contracts/Brackets.sol#2)
        - ^0.8.4 (jobs/bracket/brackets/brownie/contracts/Config.sol#3)
        - ^0.8.4 (jobs/bracket/brackets/brownie/contracts/IPricing.sol#3)
        - ^0.8.4 (jobs/bracket/brackets/brownie/contracts/Offers.sol#3)
        - ^0.8.4 (jobs/bracket/brackets/brownie/contracts/StructLibx.sol#3)
        - ^0.8.4 (jobs/bracket/brackets/brownie/contracts/UNFT.sol#2)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used

ReentrancyGuardUpgradeable.nonReentrant() (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/security/ReentrancyGuardUpgradeable.sol#55-67) has costly operations inside a loop:
        - _status = _ENTERED (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/security/ReentrancyGuardUpgradeable.sol#60)
ReentrancyGuardUpgradeable.nonReentrant() (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/security/ReentrancyGuardUpgradeable.sol#55-67) has costly operations inside a loop:
        - _status = _NOT_ENTERED (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/security/ReentrancyGuardUpgradeable.sol#66)
Brackets.autoClaim(uint16) (jobs/bracket/brackets/brownie/contracts/Brackets.sol#394-413) has costly operations inside a loop:
        - autoClaimPtr += 1 (jobs/bracket/brackets/brownie/contracts/Brackets.sol#409)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#costly-operations-inside-a-loop

AddressUpgradeable.functionCall(address,bytes) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/AddressUpgradeable.sol#85-87) is never used and should be removed
AddressUpgradeable.functionCall(address,bytes,string) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/AddressUpgradeable.sol#95-101) is never used and should be removed
AddressUpgradeable.functionCallWithValue(address,bytes,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/AddressUpgradeable.sol#114-120) is never used and should be removed
AddressUpgradeable.functionCallWithValue(address,bytes,uint256,string) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/AddressUpgradeable.sol#128-139) is never used and should be removed
AddressUpgradeable.functionStaticCall(address,bytes) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/AddressUpgradeable.sol#147-149) is never used and should be removed
AddressUpgradeable.functionStaticCall(address,bytes,string) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/AddressUpgradeable.sol#157-166) is never used and should be removed
AddressUpgradeable.sendValue(address,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/AddressUpgradeable.sol#60-65) is never used and should be removed
AddressUpgradeable.verifyCallResult(bool,bytes,string) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/AddressUpgradeable.sol#174-194) is never used and should be removed
ContextUpgradeable._msgData() (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/ContextUpgradeable.sol#18-19) is never used and should be removed
ContextUpgradeable.__Context_init() (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/ContextUpgradeable.sol#21-22) is never used and should be removed
ContextUpgradeable.__Context_init_unchained() (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/ContextUpgradeable.sol#23-29) is never used and should be removed
CountersUpgradeable.decrement(CountersUpgradeable.Counter) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/CountersUpgradeable.sol#32-38) is never used and should be removed
CountersUpgradeable.reset(CountersUpgradeable.Counter) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/CountersUpgradeable.sol#40-42) is never used and should be removed
ERC165Upgradeable._ERC165_init() (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/introspection/ERC165Upgradeable.sol#24-25) is never used and should be removed
ERC165Upgradeable.__ERC165_init_unchained() (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/introspection/ERC165Upgradeable.sol#27-28) is never used and should be removed
ERC721EnumerableUpgradeable.__ERC721Enumerable_init() (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/token/ERC721/extensions/ERC721EnumerableUpgradeable.sol#16-17) is never used and should be removed
ERC721EnumerableUpgradeable.__ERC721Enumerable_init_unchained() (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/token/ERC721/extensions/ERC721EnumerableUpgradeable.sol#19-20) is never used and should be removed
ERC721Upgradeable._burn(uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#308-322) is never used and should be removed
Initializable._disableInitializers() (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/proxy/utils/Initializable.sol#131-137) is never used and should be removed
MathUpgradeable.average(uint256,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#34-37) is never used and should be removed
MathUpgradeable.ceilDiv(uint256,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#45-48) is never used and should be removed
MathUpgradeable.max(uint256,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#19-21) is never used and should be removed
MathUpgradeable.mulDiv(uint256,uint256,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#55-135) is never used and should be removed
MathUpgradeable.mulDiv(uint256,uint256,uint256,MathUpgradeable.Rounding) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#140-151) is never used and should be removed
MathUpgradeable.sqrt(uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#158-214) is never used and should be removed
MathUpgradeable.sqrt(uint256,MathUpgradeable.Rounding) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#219-223) is never used and should be removed
SafeMathUpgradeable.div(uint256,string) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/SafeMathUpgradeable.sol#191-200) is never used and should be removed
SafeMathUpgradeable.mod(uint256,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/SafeMathUpgradeable.sol#131-153) is never used and should be removed
SafeMathUpgradeable.mod(uint256,uint256,string) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/SafeMathUpgradeable.sol#217-226) is never used and should be removed
SafeMathUpgradeable.sub(uint256,string) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/SafeMathUpgradeable.sol#168-177) is never used and should be removed
SafeMathUpgradeable.tryAdd(uint256,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/SafeMathUpgradeable.sol#22-28) is never used and should be removed
SafeMathUpgradeable.tryDiv(uint256,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/SafeMathUpgradeable.sol#64-69) is never used and should be removed
SafeMathUpgradeable.tryMod(uint256,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/SafeMathUpgradeable.sol#76-81) is never used and should be removed
SafeMathUpgradeable.tryMul(uint256,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/SafeMathUpgradeable.sol#47-57) is never used and should be removed
SafeMathUpgradeable.trySub(uint256,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/SafeMathUpgradeable.sol#35-40) is never used and should be removed
StringsUpgradeable.toHexString(address) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/StringsUpgradeable.sol#72-74) is never used and should be removed
StringsUpgradeable.toHexString(uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/StringsUpgradeable.sol#41-52) is never used and should be removed
StringsUpgradeable.toHexString(uint256,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/StringsUpgradeable.sol#57-67) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.8.0 (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/access/OwnableUpgradeable.sol#4) allows old versions
Pragma version^0.8.2 (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/proxy/utils/Initializable.sol#4) allows old versions
Pragma version^0.8.0 (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/security/PausableUpgradeable.sol#4) allows old versions
Pragma version^0.8.0 (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/security/ReentrancyGuardUpgradeable.sol#4) allows old versions
Pragma version^0.8.0 (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#4) allows old versions
Pragma version^0.8.0 (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/token/ERC721/IERC721ReceiverUpgradeable.sol#4) allows old versions
Pragma version^0.8.0 (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/token/ERC721/IERC721Upgradeable.sol#4) allows old versions
Pragma version^0.8.0 (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/token/ERC721/extensions/ERC721EnumerableUpgradeable.sol#4) allows old versions
Pragma version^0.8.0 (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/token/ERC721/extensions/IERC721EnumerableUpgradeable.sol#4) allows old versions
Pragma version^0.8.0 (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/token/ERC721/extensions/IERC721MetadataUpgradeable.sol#4) allows old versions
Pragma version^0.8.1 (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/AddressUpgradeable.sol#4) allows old versions
Pragma version^0.8.0 (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/ContextUpgradeable.sol#4) allows old versions
Pragma version^0.8.0 (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/CountersUpgradeable.sol#4) allows old versions
Pragma version^0.8.0 (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/StringsUpgradeable.sol#4) allows old versions
Pragma version^0.8.0 (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/introspection/ERC165Upgradeable.sol#4) allows old versions
Pragma version^0.8.0 (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/introspection/IERC165Upgradeable.sol#4) allows old versions
Pragma version^0.8.0 (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#4) allows old versions
Pragma version^0.8.0 (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/SafeMathUpgradeable.sol#4) allows old versions
Pragma version^0.8.0 (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts/token/ERC20/IERC20.sol#4) allows old versions
solc-0.8.16 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in Address.sendValue(address,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/AddressUpgradeable.sol#60-65):
        - (success) = recipient.call{value: amount}() (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/AddressUpgradeable.sol#63)
Low level call in AddressUpgradeable.functionCallWithValue(address,bytes,uint256,string) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/AddressUpgradeable.sol#128-139):
        - (success,returndata) = target.call{value: value}(data) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/AddressUpgradeable.sol#137)
Low level call in AddressUpgradeable.functionStaticCall(address,bytes,string) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/AddressUpgradeable.sol#157-166):
        - (success,returndata) = target.staticcall(data) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/AddressUpgradeable.sol#164)
Low level call in Brackets.buyPolicy(address,address,uint8,uint256,uint256) (jobs/bracket/brackets/brownie/contracts/Brackets.sol#172-294):
        - (success2,data2) = address(ibnftl).call(abi.encodeWithSignature(mintNFTinsured(address),msg.sender)) (jobs/bracket/brackets/brownie/contracts/Brackets.sol#230-241)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
```

```
Function OwnableUpgradeable._Ownable_init() (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/access/OwnableUpgradeable.sol#29-31) is not in mixedCase
Function OwnableUpgradeable._Ownable_init_unchained() (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/access/OwnableUpgradeable.sol#33-35) is not in mixedCase
Variable OwnableUpgradeable.__gap (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/access/OwnableUpgradeable.sol#94) is not in mixedCase
Function PausableUpgradeable.__Pausable_init() (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/security/PausableUpgradeable.sol#34-36) is not in mixedCase
Function PausableUpgradeable.__Pausable_init_unchained() (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/security/PausableUpgradeable.sol#38-40) is not in mixedCase
Variable PausableUpgradeable.__gap (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/security/PausableUpgradeable.sol#116) is not in mixedCase
Function ReentrancyGuardUpgradeable.__ReentrancyGuard_init() (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/security/ReentrancyGuardUpgradeable.sol#40-42) is not in mixedCase
Function ReentrancyGuardUpgradeable.__ReentrancyGuard_init_unchained() (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/security/ReentrancyGuardUpgradeable.sol#44-46) is not in mixedCase
Variable ReentrancyGuardUpgradeable.__gap (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/security/ReentrancyGuardUpgradeable.sol#74) is not in mixedCase
Function ERC721Upgradeable.__ERC721_init(string,string) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#45-47) is not in mixedCase
Function ERC721Upgradeable.__ERC721_init_unchained(string,string) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#49-52) is not in mixedCase
Variable ERC721Upgradeable.__gap (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#465) is not in mixedCase
Function ERC721EnumerableUpgradeable.__ERC721Enumerable_init() (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/token/ERC721/extensions/ERC721EnumerableUpgradeable.sol#16-17) is not in mixedCase
Function ERC721EnumerableUpgradeable.__ERC721Enumerable_init_unchained() (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/token/ERC721/extensions/ERC721EnumerableUpgradeable.sol#19-20) is not in mixedCase
Variable ERC721EnumerableUpgradeable.__gap (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/token/ERC721/extensions/ERC721EnumerableUpgradeable.sol#75) is not in mixedCase
Function ContextUpgradeable.__Context_init() (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/ContextUpgradeable.sol#18-19) is not in mixedCase
Function ContextUpgradeable.__Context_init_unchained() (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/ContextUpgradeable.sol#21-22) is not in mixedCase
Variable ContextUpgradeable.__gap (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/ContextUpgradeable.sol#36) is not in mixedCase
Function ERC165Upgradeable.__ERC165_init() (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/introspection/ERC165Upgradeable.sol#24-25) is not in mixedCase
Function ERC165Upgradeable.__ERC165_init_unchained() (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/introspection/ERC165Upgradeable.sol#27-28) is not in mixedCase
Variable ERC165Upgradeable.__gap (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/introspection/ERC165Upgradeable.sol#41) is not in mixedCase
Struct Brackets.buyLocals (jobs/bracket/brackets/brownie/contracts/Brackets.sol#151-160) is not in CapWords
Struct Brackets.claimLocals (jobs/bracket/brackets/brownie/contracts/Brackets.sol#206-307) is not in CapWords
Parameter Brackets.initialize(address,address,address,address,address)._ibnft (jobs/bracket/bracket/brownie/contracts/Brackets.sol#86) is not in mixedCase
Parameter Brackets.initialize(address,address,address,address,address)._usdc (jobs/bracket/bracket/brownie/contracts/Brackets.sol#87) is not in mixedCase
Parameter Brackets.initialize(address,address,address,address,address)._pricing (jobs/bracket/bracket/brownie/contracts/Brackets.sol#88) is not in mixedCase
Parameter Brackets.initialize(address,address,address,address,address)._offers (jobs/bracket/bracket/brownie/contracts/Brackets.sol#89) is not in mixedCase
Parameter Brackets.initialize(address,address,address,address,address)._config (jobs/bracket/bracket/brownie/contracts/Brackets.sol#90) is not in mixedCase
Parameter Brackets.getWithdrawAmts(address)._acct (jobs/bracket/brackets/brownie/contracts/Brackets.sol#114) is not in mixedCase
Parameter Brackets.addAvailable(address,uint256)._usdcValue (jobs/bracket/bracket/brownie/contracts/Brackets.sol#125) is not in mixedCase
Parameter Brackets.reduceAvailable(address,uint256)._newValue (jobs/bracket/bracket/brownie/contracts/Brackets.sol#141) is not in mixedCase
Parameter Brackets.buyPolicy(address,address,uint8,uint256,uint256)._asset (jobs/bracket/brackets/brownie/contracts/Brackets.sol#173) is not in mixedCase
Parameter Brackets.buyPolicy(address,address,uint8,uint256,uint256)._funder (jobs/bracket/bracket/brownie/contracts/Brackets.sol#174) is not in mixedCase
Parameter Brackets.buyPolicy(address,address,uint8,uint256,uint256)._offerId (jobs/bracket/bracket/brownie/contracts/Brackets.sol#175) is not in mixedCase
Parameter Brackets.buyPolicy(address,address,uint8,uint256,uint256)._maxAssetPrice (jobs/bracket/bracket/brownie/contracts/Brackets.sol#176) is not in mixedCase
Parameter Brackets.buyPolicy(address,address,uint8,uint256,uint256)._minAssetPrice (jobs/bracket/bracket/brownie/contracts/Brackets.sol#177) is not in mixedCase
Parameter Brackets.claim(uint256)._policyNum (jobs/bracket/bracket/brownie/contracts/Brackets.sol#312) is not in mixedCase
Variable Brackets.FundersAddress (jobs/bracket/brackets/brownie/contracts/Brackets.sol#28) is not in mixedCase
Function Config.setMIN_AVAIL_NEW(uint256) (jobs/bracket/brackets/brownie/contracts/Config.sol#79-81) is not in mixedCase
Function Config.setMIN_PREM(uint256) (jobs/bracket/brackets/brownie/contracts/Config.sol#83-85) is not in mixedCase
Function Config.setPREM_COMMISN(uint256) (jobs/bracket/bracket/brownie/contracts/Config.sol#87-89) is not in mixedCase
Function Config.setCLAIM_COMMISN(uint256) (jobs/bracket/bracket/brownie/contracts/Config.sol#91-93) is not in mixedCase
Variable Config.MIN_AVAIL_NEW (jobs/bracket/brackets/brownie/contracts/Config.sol#72) is not in mixedCase
Variable Config.MIN_PREM (jobs/bracket/bracket/brownie/contracts/Config.sol#73) is not in mixedCase
Variable Config.PREM_COMMISN (jobs/bracket/bracket/brownie/contracts/Config.sol#74) is not in mixedCase
Variable Config.CLAIM_COMMISN (jobs/bracket/bracket/brownie/contracts/Config.sol#75) is not in mixedCase
Variable Config.Vers (jobs/bracket/bracket/brownie/contracts/Config.sol#76) is not in mixedCase
Parameter Offerss.initialize(address,address)._pricing (jobs/bracket/brackets/brownie/contracts/Offerss.sol#39) is not in mixedCase
Parameter Offerss.initialize(address,address)._config (jobs/bracket/brackets/brownie/contracts/Offerss.sol#39) is not in mixedCase
Variable Offerss.Offers (jobs/bracket/brackets/brownie/contracts/Offerss.sol#29) is not in mixedCase
Contract iBNFT (jobs/bracket/brackets/brownie/contracts/iBNFT.sol#13-104) is not in CapWords
Parameter iBNFT.setBrkt(address)._brkt (jobs/bracket/brackets/brownie/contracts/iBNFT.sol#40) is not in mixedCase
Variable iBNFT.Policies (jobs/bracket/brackets/brownie/contracts/iBNFT.sol#24) is not in mixedCase
Variable iBNFT._tokenIds (jobs/bracket/brackets/brownie/contracts/iBNFT.sol#26) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions


Reentrancy in Brackets.withdraw() (jobs/bracket/brackets/brownie/contracts/Brackets.sol#416-429):
        External calls:
        - address(msg.sender).transfer(amountETH) (jobs/bracket/brackets/brownie/contracts/Brackets.sol#422)
        State variables written after the call(s):
        - pendingUSDwithdrawals[msg.sender] = 0 (jobs/bracket/brackets/brownie/contracts/Brackets.sol#425)
        Event emitted after the call(s):
        - WithdrawEv(msg.sender,amountETH,amountUSDC) (jobs/bracket/brackets/brownie/contracts/Brackets.sol#428)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-4

ReentrancyGuardUpgradeable.__gap (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/security/ReentrancyGuardUpgradeable.sol#74) is never used in Brackets (jobs/bracket/brackets/brownie/contracts/Brackets.sol#10-430)
OwnableUpgradeable.__gap (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/access/OwnableUpgradeable.sol#94) is never used in Config (jobs/bracket/brackets/brownie/contracts/Config.sol#1-94)
ReentrancyGuardUpgradeable.__gap (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/security/ReentrancyGuardUpgradeable.sol#74) is never used in Offerss (jobs/bracket/brackets/brownie/contracts/Offerss.sol#15-94)
StructLibs.ETH (jobs/bracket/brackets/brownie/contracts/StructLibs.sol#13) is never used in StructLibs (jobs/bracket/brackets/brownie/contracts/StructLibs.sol#10-56)
ReentrancyGuardUpgradeable.__gap (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/security/ReentrancyGuardUpgradeable.sol#74) is never used in iBNFT (jobs/bracket/brackets/brownie/contracts/iBNFT.sol#13-104)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variable

renounceOwnership() should be declared external:
        - OwnableUpgradeable.renounceOwnership() (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/access/OwnableUpgradeable.sol#66-68)
transferOwnership(address) should be declared external:
        - OwnableUpgradeable.transferOwnership(address) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/access/OwnableUpgradeable.sol#74-77)
name() should be declared external:
        - ERC721Upgradeable.name() (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#84-86)
symbol() should be declared external:
        - ERC721Upgradeable.symbol() (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#91-93)
tokenURI(uint256) should be declared external:
        - ERC721Upgradeable.tokenURI(uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#98-103)
approve(address,uint256) should be declared external:
        - ERC721Upgradeable.approve(address,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#117-127)
setApprovalForAll(address,bool) should be declared external:
        - ERC721Upgradeable.setApprovalForAll(address,bool) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#141-143)
safeTransferFrom(address,address,uint256) should be declared external:
        - ERC721Upgradeable.safeTransferFrom(address,address,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#169-175)
tokenOfOwnerByIndex(address,uint256) should be declared external:
        - ERC721EnumerableUpgradeable.tokenOfOwnerByIndex(address,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/token/ERC721/extensions/ERC721EnumerableUpgradeable.sol#43-46)
tokenByIndex(uint256) should be declared external:
        - ERC721EnumerableUpgradeable.tokenByIndex(uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/token/ERC721/extensions/ERC721EnumerableUpgradeable.sol#58-61)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
./jobs/bracket/brackets/brownie/contracts/Brackets.sol analyzed (25 contracts with 78 detectors), 187 result(s) found
```

Config.sol

```
MathUpgradeable.mulDiv(uint256,uint256,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#55-135) performs a multiplication on the result of a division:
        -denominator = denominator / twos (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#102)
        -inverse = (3 * denominator) ^ 2 (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#117)
MathUpgradeable.mulDiv(uint256,uint256,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#55-135) performs a multiplication on the result of a division:
        -denominator = denominator / twos (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#102)
        -inverse *= 2 - denominator * inverse (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#121)
MathUpgradeable.mulDiv(uint256,uint256,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#55-135) performs a multiplication on the result of a division:
        -denominator = denominator / twos (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#102)
        -inverse *= 2 - denominator * inverse (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#122)
MathUpgradeable.mulDiv(uint256,uint256,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#55-135) performs a multiplication on the result of a division:
        -denominator = denominator / twos (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#102)
        -inverse *= 2 - denominator * inverse (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#123)
MathUpgradeable.mulDiv(uint256,uint256,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#55-135) performs a multiplication on the result of a division:
        -denominator = denominator / twos (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#102)
        -inverse *= 2 - denominator * inverse (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#124)
MathUpgradeable.mulDiv(uint256,uint256,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#55-135) performs a multiplication on the result of a division:
        -denominator = denominator / twos (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#102)
        -inverse *= 2 - denominator * inverse (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#125)
MathUpgradeable.mulDiv(uint256,uint256,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#55-135) performs a multiplication on the result of a division:
        -denominator = denominator / twos (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#102)
        -inverse *= 2 - denominator * inverse (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#126)
MathUpgradeable.mulDiv(uint256,uint256,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#55-135) performs a multiplication on the result of a division:
        -prod0 = prod0 / twos (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#105)
        -result = prod0 * inverse (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#132)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply

AddressUpgradeable.verifyCallResult(bool,bytes,string) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/AddressUpgradeable.sol#174-194) uses assembly
        - INLINE ASM (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/AddressUpgradeable.sol#186-189)
MathUpgradeable.mulDiv(uint256,uint256,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#55-135) uses assembly
        - INLINE ASM (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#66-70)
        - INLINE ASM (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#86-93)
        - INLINE ASM (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#108-109)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

Different versions of Solidity are used:
        - Version used: ['^0.8.0', '^0.8.1', '^0.8.2', '^0.8.4']
        - ^0.8.0 (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/access/OwnableUpgradeable.sol#4)
        - ^0.8.2 (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/proxy/utils/Initializable.sol#4)
        - ^0.8.1 (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/AddressUpgradeable.sol#4)
        - ^0.8.0 (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/ContextUpgradeable.sol#4)
        - ^0.8.0 (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#4)
        - ^0.8.0 (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/SafeMathUpgradeable.sol#4)
        - ^0.8.4 (jobs/bracket/brackets/brownie/contracts/Config.sol#3)
        - ^0.8.4 (jobs/bracket/brackets/brownie/contracts/StructLibs.sol#3)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used
MathUpgradeable.mulDiv(uint256,uint256,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#55-135) performs a multiplication on the result of a division:
        -denominator = denominator / twos (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#102)
        -inverse = (3 * denominator) ^ 2 (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#117)
MathUpgradeable.mulDiv(uint256,uint256,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#55-135) performs a multiplication on the result of a division:
        -denominator = denominator / twos (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#102)
        -inverse *= 2 - denominator * inverse (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#121)
MathUpgradeable.mulDiv(uint256,uint256,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#55-135) performs a multiplication on the result of a division:
        -denominator = denominator / twos (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#102)
        -inverse *= 2 - denominator * inverse (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#122)
MathUpgradeable.mulDiv(uint256,uint256,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#55-135) performs a multiplication on the result of a division:
        -denominator = denominator / twos (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#102)
        -inverse *= 2 - denominator * inverse (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#123)
MathUpgradeable.mulDiv(uint256,uint256,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#55-135) performs a multiplication on the result of a division:
        -denominator = denominator / twos (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#102)
        -inverse *= 2 - denominator * inverse (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#124)
MathUpgradeable.mulDiv(uint256,uint256,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#55-135) performs a multiplication on the result of a division:
        -denominator = denominator / twos (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#102)
        -inverse *= 2 - denominator * inverse (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#125)
MathUpgradeable.mulDiv(uint256,uint256,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#55-135) performs a multiplication on the result of a division:
        -denominator = denominator / twos (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#102)
        -inverse *= 2 - denominator * inverse (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#126)
MathUpgradeable.mulDiv(uint256,uint256,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#55-135) performs a multiplication on the result of a division:
        -prod0 = prod0 / twos (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#105)
        -result = prod0 * inverse (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#132)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply

AddressUpgradeable.verifyCallResult(bool,bytes,string) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/AddressUpgradeable.sol#174-194) uses assembly
        - INLINE ASM (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/AddressUpgradeable.sol#186-189)
MathUpgradeable.mulDiv(uint256,uint256,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#55-135) uses assembly
        - INLINE ASM (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#66-70)
        - INLINE ASM (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#86-93)
        - INLINE ASM (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#108-109)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

Different versions of Solidity are used:
        - Version used: ['^0.8.0', '^0.8.1', '^0.8.2', '^0.8.4']
        - ^0.8.0 (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/access/OwnableUpgradeable.sol#4)
        - ^0.8.2 (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/proxy/utils/Initializable.sol#4)
        - ^0.8.1 (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/AddressUpgradeable.sol#4)
        - ^0.8.0 (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/ContextUpgradeable.sol#4)
        - ^0.8.0 (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#4)
        - ^0.8.0 (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/SafeMathUpgradeable.sol#4)
        - ^0.8.4 (jobs/bracket/brackets/brownie/contracts/Config.sol#3)
        - ^0.8.4 (jobs/bracket/brackets/brownie/contracts/StructLibs.sol#3)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used
```

```
Different versions of Solidity are used:
        - Version used: ['^0.8.0', '^0.8.1', '^0.8.2', '^0.8.4']
        - ^0.8.0 (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/access/OwnableUpgradeable.sol#4)
        - ^0.8.2 (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/proxy/utils/Initializable.sol#4)
        - ^0.8.1 (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/AddressUpgradeable.sol#4)
        - ^0.8.0 (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/ContextUpgradeable.sol#4)
        - ^0.8.0 (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#4)
        - ^0.8.0 (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/SafeMathUpgradeable.sol#4)
        - ^0.8.4 (jobs/bracket/brackets/brownie/contracts/Config.sol#3)
        - ^0.8.4 (jobs/bracket/brackets/brownie/contracts/StructLibx.sol#3)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used

AddressUpgradeable.functionCall(address,bytes) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/AddressUpgradeable.sol#85-87) is never used and should be removed
AddressUpgradeable.functionCall(address,bytes,string) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/AddressUpgradeable.sol#95-101) is never used and should be removed
AddressUpgradeable.functionCallWithValue(address,bytes,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/AddressUpgradeable.sol#114-120) is never used and should be removed
AddressUpgradeable.functionCallWithValue(address,bytes,uint256,string) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/AddressUpgradeable.sol#126-139) is never used and should be removed
AddressUpgradeable.functionStaticCall(address,bytes) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/AddressUpgradeable.sol#147-149) is never used and should be removed
AddressUpgradeable.functionStaticCall(address,bytes,string) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/AddressUpgradeable.sol#157-166) is never used and should be removed
AddressUpgradeable.sendValue(address,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/AddressUpgradeable.sol#60-65) is never used and should be removed
AddressUpgradeable.verifyCallResult(bool,bytes,string) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/AddressUpgradeable.sol#174-194) is never used and should be removed
ContextUpgradeable._Context_init() (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/ContextUpgradeable.sol#18-19) is never used and should be removed
ContextUpgradeable.__Context_init_unchained() (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/ContextUpgradeable.sol#21-22) is never used and should be removed
ContextUpgradeable._msgData() (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/ContextUpgradeable.sol#27-29) is never used and should be removed
Initializable._disableInitializers() (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/proxy/utils/Initializable.sol#131-137) is never used and should be removed
MathUpgradeable.average(uint256,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#34-37) is never used and should be removed
MathUpgradeable.ceilDiv(uint256,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#45-48) is never used and should be removed
MathUpgradeable.max(uint256,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#19-21) is never used and should be removed
MathUpgradeable.min(uint256,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#26-28) is never used and should be removed
MathUpgradeable.mulDiv(uint256,uint256,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#55-135) is never used and should be removed
MathUpgradeable.mulDiv(uint256,uint256,uint256,MathUpgradeable.Rounding) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#140-151) is never used and should be removed
MathUpgradeable.sqrt(uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#158-214) is never used and should be removed
MathUpgradeable.sqrt(uint256,MathUpgradeable.Rounding) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#219-225) is never used and should be removed
SafeMathUpgradeable.add(uint256,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/SafeMathUpgradeable.sol#93-95) is never used and should be removed
SafeMathUpgradeable.div(uint256,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/SafeMathUpgradeable.sol#135-137) is never used and should be removed
SafeMathUpgradeable.div(uint256,uint256,string) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/SafeMathUpgradeable.sol#191-200) is never used and should be removed
SafeMathUpgradeable.mod(uint256,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/SafeMathUpgradeable.sol#151-153) is never used and should be removed
SafeMathUpgradeable.mod(uint256,uint256,string) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/SafeMathUpgradeable.sol#217-226) is never used and should be removed
SafeMathUpgradeable.mul(uint256,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/SafeMathUpgradeable.sol#121-123) is never used and should be removed
SafeMathUpgradeable.sub(uint256,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/SafeMathUpgradeable.sol#107-109) is never used and should be removed
SafeMathUpgradeable.sub(uint256,uint256,string) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/SafeMathUpgradeable.sol#168-177) is never used and should be removed
SafeMathUpgradeable.tryAdd(uint256,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/SafeMathUpgradeable.sol#22-28) is never used and should be removed
SafeMathUpgradeable.trySub(uint256,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/SafeMathUpgradeable.sol#64-69) is never used and should be removed
SafeMathUpgradeable.tryMod(uint256,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/SafeMathUpgradeable.sol#76-81) is never used and should be removed
SafeMathUpgradeable.tryMul(uint256,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/SafeMathUpgradeable.sol#47-57) is never used and should be removed
SafeMathUpgradeable.tryDiv(uint256,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/SafeMathUpgradeable.sol#35-40) is never used and should be removed
StructLibx.multFactor(uint256,uint256) (jobs/bracket/brackets/brownie/contracts/StructLibx.sol#49-55) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.8.0 (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/access/OwnableUpgradeable.sol#4) allows old versions
Pragma version^0.8.2 (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/proxy/utils/Initializable.sol#4) allows old versions
Pragma version^0.8.1 (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/AddressUpgradeable.sol#4) allows old versions
Pragma version^0.8.0 (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/ContextUpgradeable.sol#4) allows old versions
Pragma version^0.8.0 (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#4) allows old versions
Pragma version^0.8.0 (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/SafeMathUpgradeable.sol#4) allows old versions
solc-0.8.16 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in AddressUpgradeable.sendValue(address,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/AddressUpgradeable.sol#60-65):
        - (success) = recipient.call{value: amount}() (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/AddressUpgradeable.sol#63)
Low level call in AddressUpgradeable.functionCallWithValue(address,bytes,uint256,string) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/AddressUpgradeable.sol#128-139):
        - (success,returndata) = target.call{value: value}(data) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/AddressUpgradeable.sol#137)
Low level call in AddressUpgradeable.functionStaticCall(address,bytes,string) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/AddressUpgradeable.sol#157-166):
        - (success,returndata) = target.staticcall(data) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/AddressUpgradeable.sol#164)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Function OwnableUpgradeable.__Ownable_init() (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/access/OwnableUpgradeable.sol#29-31) is not in mixedCase
Function OwnableUpgradeable.__Ownable_init_unchained() (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/access/OwnableUpgradeable.sol#33-35) is not in mixedCase
Variable OwnableUpgradeable.__gap (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/access/OwnableUpgradeable.sol#94) is not in mixedCase
Function ContextUpgradeable._Context_init() (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/ContextUpgradeable.sol#18-19) is not in mixedCase
Function ContextUpgradeable.__Context_init_unchained() (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/ContextUpgradeable.sol#21-22) is not in mixedCase
Variable ContextUpgradeable.__gap (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/ContextUpgradeable.sol#36) is not in mixedCase
Function Config.setMIN_AVAIL_NEW(uint256) (jobs/bracket/brackets/brownie/contracts/Config.sol#79-81) is not in mixedCase
Function Config.setMIN_PREM(uint256) (jobs/bracket/brackets/brownie/contracts/Config.sol#83-85) is not in mixedCase
Function Config.setPREM_COMMISH(uint256) (jobs/bracket/brackets/brownie/contracts/Config.sol#87-89) is not in mixedCase
Function Config.setCLAIM_COMMISH(uint256) (jobs/bracket/brackets/brownie/contracts/Config.sol#91-93) is not in mixedCase
Variable Config.MIN_AVAIL_NEW (jobs/bracket/brackets/brownie/contracts/Config.sol#12) is not in mixedCase
Variable Config.MIN_PREM (jobs/bracket/brackets/brownie/contracts/Config.sol#13) is not in mixedCase
Variable Config.PREM_COMMISH (jobs/bracket/brackets/brownie/contracts/Config.sol#14) is not in mixedCase
Variable Config.CLAIM_COMMISH (jobs/bracket/brackets/brownie/contracts/Config.sol#15) is not in mixedCase
Variable Config.Vers (jobs/bracket/brackets/brownie/contracts/Config.sol#16) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

OwnableUpgradeable.__gap (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/access/OwnableUpgradeable.sol#94) is never used in Config (jobs/bracket/brackets/brownie/contracts/Config.sol#11-94)
StructLibx.ETH (jobs/bracket/brackets/brownie/contracts/StructLibx.sol#13) is never used in StructLibx (jobs/bracket/brackets/brownie/contracts/StructLibx.sol#5-56)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variable

renounceOwnership() should be declared external:
        - OwnableUpgradeable.renounceOwnership() (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/access/OwnableUpgradeable.sol#66-68)
transferOwnership(address) should be declared external:
        - OwnableUpgradeable.transferOwnership(address) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/access/OwnableUpgradeable.sol#74-77)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
./jobs/bracket/brackets/brownie/contracts/Config.sol analyzed (8 contracts with 78 detectors), 74 result(s) found
```

iBNFT.sol

```
MathUpgradeable.mulDiv(uint256,uint256,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#55-135) performs a multiplication on the result of a division:
        -denominator = denominator / twos (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#102)
        -inverse = (3 * denominator) ^ 2 (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#117)
MathUpgradeable.mulDiv(uint256,uint256,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#55-135) performs a multiplication on the result of a division:
        -denominator = denominator / twos (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#102)
        -inverse *= 2 - denominator * inverse (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#121)
MathUpgradeable.mulDiv(uint256,uint256,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#55-135) performs a multiplication on the result of a division:
        -denominator = denominator / twos (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#102)
        -inverse *= 2 - denominator * inverse (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#122)
MathUpgradeable.mulDiv(uint256,uint256,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#55-135) performs a multiplication on the result of a division:
        -denominator = denominator / twos (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#102)
        -inverse *= 2 - denominator * inverse (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#123)
MathUpgradeable.mulDiv(uint256,uint256,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#55-135) performs a multiplication on the result of a division:
        -denominator = denominator / twos (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#102)
        -inverse *= 2 - denominator * inverse (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#124)
MathUpgradeable.mulDiv(uint256,uint256,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#55-135) performs a multiplication on the result of a division:
        -denominator = denominator / twos (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#102)
        -inverse *= 2 - denominator * inverse (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#125)
MathUpgradeable.mulDiv(uint256,uint256,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#55-135) performs a multiplication on the result of a division:
        -denominator = denominator / twos (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#102)
        -inverse *= 2 - denominator * inverse (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#126)
MathUpgradeable.mulDiv(uint256,uint256,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#55-135) performs a multiplication on the result of a division:
        -prod0 = prod0 / twos (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#105)
        -result = prod0 * inverse (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#132)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply

ERC721Upgradeable._checkOnERC721Received(address,address,uint256,bytes) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#399-421) ignores return value by IERC721ReceiverUpgradeable(to).onERC721Received(_msgSender(),from,tokenId,data) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#406-417)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return

iBNFT.setBrkt(address) (jobs/bracket/brackets/brownie/contracts/iBNFT.sol#40-42) should emit an event for:
        - brkt = _brkt (jobs/bracket/brackets/brownie/contracts/iBNFT.sol#41)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-access-control

iBNFT.setBrkt(address)._brkt (jobs/bracket/brackets/brownie/contracts/iBNFT.sol#40) lacks a zero-check on :
        - brkt = _brkt (jobs/bracket/brackets/brownie/contracts/iBNFT.sol#41)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

Variable 'ERC721Upgradeable._checkOnERC721Received(address,address,uint256,bytes).retval (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#406)' in ERC721Upgradeable._checkOnERC721Received(address,address,uint256,bytes) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#399-421) potentially used before declaration: retval == IERC721ReceiverUpgradeable.onERC721Received.selector (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#407)
Variable 'ERC721Upgradeable._checkOnERC721Received(address,address,uint256,bytes).reason (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#408)' in ERC721Upgradeable._checkOnERC721Received(address,address,uint256,bytes) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#399-421) potentially used before declaration: reason.length == 0 (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#409)
Variable 'ERC721Upgradeable._checkOnERC721Received(address,address,uint256,bytes).reason (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#408)' in ERC721Upgradeable._checkOnERC721Received(address,address,uint256,bytes) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#399-421) potentially used before declaration: revert(uint256,uint256)(32 + reason,mload(uint256)(reason)) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#414)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#pre-declaration-usage-of-local-variables

ERC721Upgradeable._checkOnERC721Received(address,address,uint256,bytes) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#399-421) uses assembly
        - INLINE ASM (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#413-415)
AddressUpgradeable.verifyCallResult(bool,bytes,string) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/AddressUpgradeable.sol#174-194) uses assembly
        - INLINE ASM (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/AddressUpgradeable.sol#186-189)
MathUpgradeable.mulDiv(uint256,uint256,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#55-135) uses assembly
        - INLINE ASM (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#60-70)
        - INLINE ASM (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#86-93)
        - INLINE ASM (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#100-109)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

Different versions of Solidity are used:
        - Version used: ['^0.8.0', '^0.8.1', '^0.8.2', '^0.8.4']
        - ^0.8.0 (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/access/OwnableUpgradeable.sol#4)
        - ^0.8.2 (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/proxy/utils/Initializable.sol#4)
        - ^0.8.0 (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/security/PausableUpgradeable.sol#4)
        - ^0.8.0 (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/security/ReentrancyGuardUpgradeable.sol#4)
        - ^0.8.0 (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#4)
        - ^0.8.0 (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/token/ERC721/IERC721ReceiverUpgradeable.sol#4)
        - ^0.8.0 (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/token/ERC721/IERC721Upgradeable.sol#4)
        - ^0.8.0 (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/token/ERC721/extensions/ERC721EnumerableUpgradeable.sol#4)
        - ^0.8.0 (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/token/ERC721/extensions/IERC721EnumerableUpgradeable.sol#4)
        - ^0.8.0 (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/token/ERC721/extensions/IERC721MetadataUpgradeable.sol#4)
        - ^0.8.1 (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/AddressUpgradeable.sol#4)
        - ^0.8.0 (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/ContextUpgradeable.sol#4)
        - ^0.8.0 (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/CountersUpgradeable.sol#4)
        - ^0.8.0 (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/StringsUpgradeable.sol#4)
        - ^0.8.0 (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/introspection/ERC165Upgradeable.sol#4)
        - ^0.8.0 (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/introspection/IERC165Upgradeable.sol#4)
        - ^0.8.0 (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#4)
        - ^0.8.0 (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/SafeMathUpgradeable.sol#4)
        - ^0.8.4 (jobs/bracket/brackets/brownie/contracts/StructLibx.sol#3)
        - ^0.8.4 (jobs/bracket/brackets/brownie/contracts/iBNFT.sol#2)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used
```

```
AddressUpgradeable.functionCall(address,bytes) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/AddressUpgradeable.sol#85-87) is never used and should be removed
AddressUpgradeable.functionCall(address,bytes,string) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/AddressUpgradeable.sol#95-101) is never used and should be removed
AddressUpgradeable.functionCallWithValue(address,bytes,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/AddressUpgradeable.sol#114-120) is never used and should be removed
AddressUpgradeable.functionCallWithValue(address,bytes,uint256,string) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/AddressUpgradeable.sol#128-139) is never used and should be removed
AddressUpgradeable.functionStaticCall(address,bytes) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/AddressUpgradeable.sol#147-149) is never used and should be removed
AddressUpgradeable.functionStaticCall(address,bytes,string) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/AddressUpgradeable.sol#157-160) is never used and should be removed
AddressUpgradeable.sendValue(address,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/AddressUpgradeable.sol#60-65) is never used and should be removed
AddressUpgradeable.verifyCallResult(bool,bytes,string) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/AddressUpgradeable.sol#174-194) is never used and should be removed
ContextUpgradeable.__Context_init() (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/ContextUpgradeable.sol#18-19) is never used and should be removed
ContextUpgradeable.__Context_init_unchained() (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/ContextUpgradeable.sol#21-22) is never used and should be removed
ContextUpgradeable._msgData() (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/ContextUpgradeable.sol#27-29) is never used and should be removed
CountersUpgradeable.decrement(CountersUpgradeable.Counter) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/CountersUpgradeable.sol#32-38) is never used and should be removed
CountersUpgradeable.reset(CountersUpgradeable.Counter) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/CountersUpgradeable.sol#40-42) is never used and should be removed
ERC165Upgradeable.__ERC165_init() (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/introspection/ERC165Upgradeable.sol#24-25) is never used and should be removed
ERC165Upgradeable.__ERC165_init_unchained() (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/introspection/ERC165Upgradeable.sol#27-28) is never used and should be removed
ERC721EnumerableUpgradeable.__ERC721Enumerable_init() (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/token/ERC721/extensions/ERC721EnumerableUpgradeable.sol#16-17) is never used and should be removed
ERC721EnumerableUpgradeable.__ERC721Enumerable_init_unchained() (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/token/ERC721/extensions/ERC721EnumerableUpgradeable.sol#19-20) is never used and should be removed
ERC721Upgradeable._burn(uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#308-322) is never used and should be removed
Initializable._disableInitializers() (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/proxy/utils/Initializable.sol#131-137) is never used and should be removed
MathUpgradeable.average(uint256,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#34-37) is never used and should be removed
MathUpgradeable.ceilDiv(uint256,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#45-48) is never used and should be removed
MathUpgradeable.max(uint256,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#19-21) is never used and should be removed
MathUpgradeable.min(uint256,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#26-28) is never used and should be removed
MathUpgradeable.mulDiv(uint256,uint256,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#55-135) is never used and should be removed
MathUpgradeable.mulDiv(uint256,uint256,uint256,MathUpgradeable.Rounding) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#140-151) is never used and should be removed
MathUpgradeable.sqrt(uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#158-214) is never used and should be removed
MathUpgradeable.sqrt(uint256,MathUpgradeable.Rounding) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#219-225) is never used and should be removed
SafeMathUpgradeable.add(uint256,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/SafeMathUpgradeable.sol#93-95) is never used and should be removed
SafeMathUpgradeable.div(uint256,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/SafeMathUpgradeable.sol#133-137) is never used and should be removed
SafeMathUpgradeable.div(uint256,uint256,string) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/SafeMathUpgradeable.sol#191-208) is never used and should be removed
SafeMathUpgradeable.mod(uint256,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/SafeMathUpgradeable.sol#151-153) is never used and should be removed
SafeMathUpgradeable.mod(uint256,uint256,string) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/SafeMathUpgradeable.sol#217-226) is never used and should be removed
SafeMathUpgradeable.mul(uint256,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/SafeMathUpgradeable.sol#121-123) is never used and should be removed
SafeMathUpgradeable.sub(uint256,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/SafeMathUpgradeable.sol#107-109) is never used and should be removed
SafeMathUpgradeable.sub(uint256,uint256,string) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/SafeMathUpgradeable.sol#168-177) is never used and should be removed
SafeMathUpgradeable.tryAdd(uint256,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/SafeMathUpgradeable.sol#22-26) is never used and should be removed
SafeMathUpgradeable.trySub(uint256,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/SafeMathUpgradeable.sol#64-69) is never used and should be removed
SafeMathUpgradeable.tryMod(uint256,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/SafeMathUpgradeable.sol#76-81) is never used and should be removed
SafeMathUpgradeable.tryMul(uint256,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/SafeMathUpgradeable.sol#47-57) is never used and should be removed
SafeMathUpgradeable.trySub(uint256,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/SafeMathUpgradeable.sol#35-40) is never used and should be removed
StringsUpgradeable.toHexString(address) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/StringsUpgradeable.sol#72-74) is never used and should be removed
StringsUpgradeable.toHexString(uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/StringsUpgradeable.sol#41-52) is never used and should be removed
StringsUpgradeable.toHexString(uint256,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/StringsUpgradeable.sol#57-67) is never used and should be removed
StructLib.multFactor(uint256,uint256) (jobs/bracket/brackets/brownie/contracts/StructLibs.sol#49-55) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.8.0 (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/access/OwnableUpgradeable.sol#4) allows old versions
Pragma version^0.8.2 (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/proxy/utils/Initializable.sol#4) allows old versions
Pragma version^0.8.0 (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/security/PausableUpgradeable.sol#4) allows old versions
Pragma version^0.8.0 (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/security/ReentrancyGuardUpgradeable.sol#4) allows old versions
Pragma version^0.8.0 (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#4) allows old versions
Pragma version^0.8.0 (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/token/ERC721/IERC721ReceiverUpgradeable.sol#4) allows old versions
Pragma version^0.8.0 (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/token/ERC721/IERC721Upgradeable.sol#4) allows old versions
Pragma version^0.8.0 (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/token/ERC721/extensions/ERC721EnumerableUpgradeable.sol#4) allows old versions
Pragma version^0.8.0 (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/token/ERC721/extensions/IERC721EnumerableUpgradeable.sol#4) allows old versions
Pragma version^0.8.0 (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/token/ERC721/extensions/IERC721MetadataUpgradeable.sol#4) allows old versions
Pragma version^0.8.1 (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/AddressUpgradeable.sol#4) allows old versions
Pragma version^0.8.0 (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/ContextUpgradeable.sol#4) allows old versions
Pragma version^0.8.0 (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/CountersUpgradeable.sol#4) allows old versions
Pragma version^0.8.0 (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/StringsUpgradeable.sol#4) allows old versions
Pragma version^0.8.0 (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/introspection/ERC165Upgradeable.sol#4) allows old versions
Pragma version^0.8.0 (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/introspection/IERC165Upgradeable.sol#4) allows old versions
Pragma version^0.8.0 (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#4) allows old versions
Pragma version^0.8.0 (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/SafeMathUpgradeable.sol#4) allows old versions
solc-0.8.16 is not recomended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in AddressUpgradeable.sendValue(address,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/AddressUpgradeable.sol#60-65):
        - (success) = recipient.call{value: amount}() (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/AddressUpgradeable.sol#63)
Low level call in AddressUpgradeable.functionCallWithValue(address,bytes,uint256,string) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/AddressUpgradeable.sol#128-139):
        - (success,returndata) = target.call{value: value}(data) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/AddressUpgradeable.sol#137)
Low level call in AddressUpgradeable.functionStaticCall(address,bytes,string) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/AddressUpgradeable.sol#157-166):
        - (success,returndata) = target.staticcall(data) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/AddressUpgradeable.sol#164)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Function OwnableUpgradeable.__Ownable_init() (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/access/OwnableUpgradeable.sol#29-31) is not in mixedCase
Function OwnableUpgradeable.__Ownable_init_unchained() (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/access/OwnableUpgradeable.sol#33-35) is not in mixedCase
Variable OwnableUpgradeable.__gap (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/access/OwnableUpgradeable.sol#94) is not in mixedCase
Function PausableUpgradeable.__Pausable_init() (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/security/PausableUpgradeable.sol#34-36) is not in mixedCase
Function PausableUpgradeable.__Pausable_init_unchained() (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/security/PausableUpgradeable.sol#38-40) is not in mixedCase
Variable PausableUpgradeable.__gap (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/security/PausableUpgradeable.sol#116) is not in mixedCase
Function ReentrancyGuardUpgradeable.__ReentrancyGuard_init() (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/security/ReentrancyGuardUpgradeable.sol#40-42) is not in mixedCase
Function ReentrancyGuardUpgradeable.__ReentrancyGuard_init_unchained() (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/security/ReentrancyGuardUpgradeable.sol#44-46) is not in mixedCase
Variable ReentrancyGuardUpgradeable.__gap (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/security/ReentrancyGuardUpgradeable.sol#74) is not in mixedCase
Function ERC721Upgradeable.__ERC721_init(string,string) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#45-47) is not in mixedCase
Function ERC721Upgradeable.__ERC721_init_unchained(string,string) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#49-52) is not in mixedCase
Variable ERC721Upgradeable.__gap (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#465) is not in mixedCase
Function ERC721EnumerableUpgradeable.__ERC721Enumerable_init() (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/token/ERC721/extensions/ERC721EnumerableUpgradeable.sol#16-17) is not in mixedCase
Function ERC721EnumerableUpgradeable.__ERC721Enumerable_init_unchained() (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/token/ERC721/extensions/ERC721EnumerableUpgradeable.sol#19-20) is not in mixedCase
Variable ERC721EnumerableUpgradeable.__gap (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/token/ERC721/extensions/ERC721EnumerableUpgradeable.sol#173) is not in mixedCase
Function ContextUpgradeable.__Context_init() (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/ContextUpgradeable.sol#18-19) is not in mixedCase
Function ContextUpgradeable.__Context_init_unchained() (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/ContextUpgradeable.sol#21-22) is not in mixedCase
Variable ContextUpgradeable.__gap (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/ContextUpgradeable.sol#36) is not in mixedCase
Function ERC165Upgradeable.__ERC165_init() (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/introspection/ERC165Upgradeable.sol#24-25) is not in mixedCase
Function ERC165Upgradeable.__ERC165_init_unchained() (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/introspection/ERC165Upgradeable.sol#27-28) is not in mixedCase
Variable ERC165Upgradeable.__gap (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/introspection/ERC165Upgradeable.sol#41) is not in mixedCase
Contract IBNFT (jobs/bracket/brackets/brownie/contracts/IBNFT.sol#13-184) is not in CapWords
Parameter IBNFT.setBrkt(address)._brkt (jobs/bracket/brackets/brownie/contracts/IBNFT.sol#40) is not in mixedCase
Variable IBNFT_Policies (jobs/bracket/brackets/brownie/contracts/IBNFT.sol#24) is not in mixedCase
Variable IBNFT_tokenIds (jobs/bracket/brackets/brownie/contracts/IBNFT.sol#26) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

StructLibs.ETH (jobs/bracket/brackets/brownie/contracts/StructLibs.sol#13) is never used in StructLibs (jobs/bracket/brackets/brownie/contracts/StructLibs.sol#8-56)
ReentrancyGuardUpgradeable.__gap (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/security/ReentrancyGuardUpgradeable.sol#74) is never used in IBNFT (jobs/bracket/brackets/brownie/contracts/IBNFT.sol#13-184)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variable

renounceOwnership() should be declared external:
        - OwnableUpgradeable.renounceOwnership() (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/access/OwnableUpgradeable.sol#66-68)
transferOwnership(address) should be declared external:
        - OwnableUpgradeable.transferOwnership(address) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/access/OwnableUpgradeable.sol#74-77)
name() should be declared external:
        - ERC721Upgradeable.name() (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#84-86)
symbol() should be declared external:
        - ERC721Upgradeable.symbol() (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#91-93)
tokenURI(uint256) should be declared external:
        - ERC721Upgradeable.tokenURI(uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#98-103)
approve(address,uint256) should be declared external:
        - ERC721Upgradeable.approve(address,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#117-127)
setApprovalForAll(address,bool) should be declared external:
        - ERC721Upgradeable.setApprovalForAll(address,bool) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#141-143)
safeTransferFrom(address,address,uint256) should be declared external:
        - ERC721Upgradeable.safeTransferFrom(address,address,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#169-175)
tokenOfOwnerByIndex(address,uint256) should be declared external:
        - ERC721EnumerableUpgradeable.tokenOfOwnerByIndex(address,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/token/ERC721/extensions/ERC721EnumerableUpgradeable.sol#43-46)
tokenByIndex(uint256) should be declared external:
        - ERC721EnumerableUpgradeable.tokenByIndex(uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/token/ERC721/extensions/ERC721EnumerableUpgradeable.sol#58-61)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
./jobs/bracket/brackets/brownie/contracts/IBNFT.sol analyzed (20 contracts with 78 detectors), 121 result(s) found
```

Offersx.sol

```
MathUpgradeable.mulDiv(uint256,uint256,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#55-135) performs a multiplication on the result of a division:
        -denominator = denominator / twos (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#102)
        -inverse = (3 * denominator) ^ 2 (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#117)
MathUpgradeable.mulDiv(uint256,uint256,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#55-135) performs a multiplication on the result of a division:
        -denominator = denominator / twos (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#102)
        -inverse *= 2 - denominator * inverse (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#121)
MathUpgradeable.mulDiv(uint256,uint256,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#55-135) performs a multiplication on the result of a division:
        -denominator = denominator / twos (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#102)
        -inverse *= 2 - denominator * inverse (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#122)
MathUpgradeable.mulDiv(uint256,uint256,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#55-135) performs a multiplication on the result of a division:
        -denominator = denominator / twos (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#102)
        -inverse *= 2 - denominator * inverse (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#123)
MathUpgradeable.mulDiv(uint256,uint256,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#55-135) performs a multiplication on the result of a division:
        -denominator = denominator / twos (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#102)
        -inverse *= 2 - denominator * inverse (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#124)
MathUpgradeable.mulDiv(uint256,uint256,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#55-135) performs a multiplication on the result of a division:
        -denominator = denominator / twos (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#102)
        -inverse *= 2 - denominator * inverse (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#125)
MathUpgradeable.mulDiv(uint256,uint256,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#55-135) performs a multiplication on the result of a division:
        -denominator = denominator / twos (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#102)
        -inverse *= 2 - denominator * inverse (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#126)
MathUpgradeable.mulDiv(uint256,uint256,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#55-135) performs a multiplication on the result of a division:
        -prod0 = prod0 / twos (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#105)
        -result = prod0 * inverse (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#132)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply

Offersx.setOfferX(address,uint256[12],uint16).i (jobs/bracket/brackets/brownie/contracts/Offersx.sol#71) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables

Offersx.initialize(address,address)._pricing (jobs/bracket/brackets/brownie/contracts/Offersx.sol#39) lacks a zero-check on :
        - pricing = _pricing (jobs/bracket/brackets/brownie/contracts/Offersx.sol#44)
Offersx.initialize(address,address)._config (jobs/bracket/brackets/brownie/contracts/Offersx.sol#39) lacks a zero-check on :
        - config = _config (jobs/bracket/brackets/brownie/contracts/Offersx.sol#45)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
```

```
AddressUpgradeable.verifyCallResult(bool,bytes,string) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/AddressUpgradeable.sol#174-194) uses assembly
        - INLINE ASM (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/AddressUpgradeable.sol#186-189)
MathUpgradeable.mulDiv(uint256,uint256,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#55-135) uses assembly
        - INLINE ASM (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#66-70)
        - INLINE ASM (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#86-93)
        - INLINE ASM (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#100-109)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

Different versions of Solidity are used:
        - Version used: ['^0.8.0', '^0.8.1', '^0.8.2', '^0.8.4']
        - ^0.8.0 (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/access/OwnableUpgradeable.sol#4)
        - ^0.8.2 (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/proxy/utils/Initializable.sol#4)
        - ^0.8.0 (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/security/PausableUpgradeable.sol#4)
        - ^0.8.0 (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/security/ReentrancyGuardUpgradeable.sol#4)
        - ^0.8.1 (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/AddressUpgradeable.sol#4)
        - ^0.8.0 (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/ContextUpgradeable.sol#4)
        - ^0.8.0 (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#4)
        - ^0.8.0 (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/SafeMathUpgradeable.sol#4)
        - ^0.8.4 (jobs/bracket/brackets/brownie/contracts/Config.sol#3)
        - ^0.8.4 (jobs/bracket/brackets/brownie/contracts/Offersx.sol#3)
        - ^0.8.4 (jobs/bracket/brackets/brownie/contracts/Structlibx.sol#3)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used

AddressUpgradeable.functionCall(address,bytes) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/AddressUpgradeable.sol#85-87) is never used and should be removed
AddressUpgradeable.functionCall(address,bytes,string) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/AddressUpgradeable.sol#95-101) is never used and should be removed
AddressUpgradeable.functionCallWithValue(address,bytes,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/AddressUpgradeable.sol#114-120) is never used and should be removed
AddressUpgradeable.functionCallWithValue(address,bytes,uint256,string) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/AddressUpgradeable.sol#128-139) is never used and should be removed
AddressUpgradeable.functionStaticCall(address,bytes) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/AddressUpgradeable.sol#147-149) is never used and should be removed
AddressUpgradeable.functionStaticCall(address,bytes,string) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/AddressUpgradeable.sol#157-166) is never used and should be removed
AddressUpgradeable.sendValue(address,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/AddressUpgradeable.sol#60-63) is never used and should be removed
AddressUpgradeable.verifyCallResult(bool,bytes,string) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/AddressUpgradeable.sol#174-194) is never used and should be removed
ContextUpgradeable.__Context_init() (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/ContextUpgradeable.sol#18-19) is never used and should be removed
ContextUpgradeable.__Context_init_unchained() (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/ContextUpgradeable.sol#21-22) is never used and should be removed
ContextUpgradeable._msgData() (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/ContextUpgradeable.sol#27-29) is never used and should be removed
Initializable._disableInitializers() (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/proxy/utils/Initializable.sol#131-137) is never used and should be removed
MathUpgradeable.average(uint256,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#34-37) is never used and should be removed
MathUpgradeable.ceilDiv(uint256,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#45-48) is never used and should be removed
MathUpgradeable.max(uint256,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#19-21) is never used and should be removed
MathUpgradeable.min(uint256,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#26-28) is never used and should be removed
MathUpgradeable.mulDiv(uint256,uint256,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#55-135) is never used and should be removed
MathUpgradeable.mulDiv(uint256,uint256,uint256,MathUpgradeable.Rounding) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#140-151) is never used and should be removed
MathUpgradeable.sqrt(uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#158-214) is never used and should be removed
MathUpgradeable.sqrt(uint256,MathUpgradeable.Rounding) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#219-225) is never used and should be removed
SafeMathUpgradeable.add(uint256,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/SafeMathUpgradeable.sol#93-95) is never used and should be removed
SafeMathUpgradeable.div(uint256,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/SafeMathUpgradeable.sol#135-137) is never used and should be removed
SafeMathUpgradeable.div(uint256,uint256,string) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/SafeMathUpgradeable.sol#191-200) is never used and should be removed
SafeMathUpgradeable.mod(uint256,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/SafeMathUpgradeable.sol#151-153) is never used and should be removed
SafeMathUpgradeable.mod(uint256,uint256,string) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/SafeMathUpgradeable.sol#217-226) is never used and should be removed
SafeMathUpgradeable.mul(uint256,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/SafeMathUpgradeable.sol#121-123) is never used and should be removed
SafeMathUpgradeable.sub(uint256,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/SafeMathUpgradeable.sol#107-109) is never used and should be removed
SafeMathUpgradeable.sub(uint256,uint256,string) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/SafeMathUpgradeable.sol#168-177) is never used and should be removed
SafeMathUpgradeable.tryAdd(uint256,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/SafeMathUpgradeable.sol#22-28) is never used and should be removed
SafeMathUpgradeable.tryDiv(uint256,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/SafeMathUpgradeable.sol#64-69) is never used and should be removed
SafeMathUpgradeable.tryMod(uint256,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/SafeMathUpgradeable.sol#76-81) is never used and should be removed
SafeMathUpgradeable.tryMul(uint256,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/SafeMathUpgradeable.sol#47-57) is never used and should be removed
SafeMathUpgradeable.trySub(uint256,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/SafeMathUpgradeable.sol#35-40) is never used and should be removed
Structlibx.multFactor(uint256,uint256) (jobs/bracket/brackets/brownie/contracts/Structlibx.sol#49-55) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.8.0 (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/access/OwnableUpgradeable.sol#4) allows old versions
Pragma version^0.8.2 (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/proxy/utils/Initializable.sol#4) allows old versions
Pragma version^0.8.0 (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/security/PausableUpgradeable.sol#4) allows old versions
Pragma version^0.8.0 (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/security/ReentrancyGuardUpgradeable.sol#4) allows old versions
Pragma version^0.8.1 (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/AddressUpgradeable.sol#4) allows old versions
Pragma version^0.8.0 (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/ContextUpgradeable.sol#4) allows old versions
Pragma version^0.8.0 (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#4) allows old versions
Pragma version^0.8.0 (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/SafeMathUpgradeable.sol#4) allows old versions
solc-0.8.16 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in AddressUpgradeable.sendValue(address,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/AddressUpgradeable.sol#60-65):
        - (success) = recipient.call{value: amount}() (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/AddressUpgradeable.sol#63)
Low level call in AddressUpgradeable.functionCallWithValue(address,bytes,uint256,string) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/AddressUpgradeable.sol#128-139):
        - (success,returndata) = target.call{value: value}(data) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/AddressUpgradeable.sol#137)
Low level call in AddressUpgradeable.functionStaticCall(address,bytes,string) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/AddressUpgradeable.sol#157-166):
        - (success,returndata) = target.staticcall(data) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/AddressUpgradeable.sol#164)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Function OwnableUpgradeable.__Ownable_init() (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/access/OwnableUpgradeable.sol#29-31) is not in mixedCase
Function OwnableUpgradeable.__Ownable_init_unchained() (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/access/OwnableUpgradeable.sol#33-35) is not in mixedCase
Variable OwnableUpgradeable.__gap (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/access/OwnableUpgradeable.sol#94) is not in mixedCase
Function PausableUpgradeable.__Pausable_init() (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/security/PausableUpgradeable.sol#34-36) is not in mixedCase
Function PausableUpgradeable.__Pausable_init_unchained() (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/security/PausableUpgradeable.sol#38-40) is not in mixedCase
Variable PausableUpgradeable.__gap (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/security/PausableUpgradeable.sol#116) is not in mixedCase
Function ReentrancyGuardUpgradeable.__ReentrancyGuard_init() (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/security/ReentrancyGuardUpgradeable.sol#40-42) is not in mixedCase
Function ReentrancyGuardUpgradeable.__ReentrancyGuard_init_unchained() (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/security/ReentrancyGuardUpgradeable.sol#44-46) is not in mixedCase
Variable ReentrancyGuardUpgradeable.__gap (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/security/ReentrancyGuardUpgradeable.sol#74) is not in mixedCase
Function ContextUpgradeable.__Context_init() (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/ContextUpgradeable.sol#18-19) is not in mixedCase
Function ContextUpgradeable.__Context_init_unchained() (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/ContextUpgradeable.sol#21-22) is not in mixedCase
Variable ContextUpgradeable.__gap (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/ContextUpgradeable.sol#36) is not in mixedCase
Function Config.setMIN_AVAIL_NEW(uint256) (jobs/bracket/brackets/brownie/contracts/Config.sol#79-81) is not in mixedCase
Function Config.setMIN_PRDH(uint256) (jobs/bracket/brackets/brownie/contracts/Config.sol#83-85) is not in mixedCase
Function Config.setPREM_COMMISH(uint256) (jobs/bracket/brackets/brownie/contracts/Config.sol#87-89) is not in mixedCase
Function Config.setCLAIM_COMMISH(uint256) (jobs/bracket/brackets/brownie/contracts/Config.sol#91-93) is not in mixedCase
Variable Config.MIN_AVAIL_NEW (jobs/bracket/brackets/brownie/contracts/Config.sol#12) is not in mixedCase
Variable Config.MIN_PREM (jobs/bracket/brackets/brownie/contracts/Config.sol#13) is not in mixedCase
Variable Config.PROM_COMMISH (jobs/bracket/brackets/brownie/contracts/Config.sol#14) is not in mixedCase
Variable Config.CLAIM_COMMISH (jobs/bracket/brackets/brownie/contracts/Config.sol#15) is not in mixedCase
Variable Config.Vers (jobs/bracket/brackets/brownie/contracts/Config.sol#16) is not in mixedCase
Parameter Offersx.initialize(address,address)._pricing (jobs/bracket/brackets/brownie/contracts/Offersx.sol#39) is not in mixedCase
Parameter Offersx.initialize(address,address)._config (jobs/bracket/brackets/brownie/contracts/Offersx.sol#39) is not in mixedCase
Variable Offersx.Offers (jobs/bracket/brackets/brownie/contracts/Offersx.sol#20) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

OwnableUpgradeable.__gap (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/access/OwnableUpgradeable.sol#94) is never used in Config (jobs/bracket/brackets/brownie/contracts/Config.sol#11-94)
ReentrancyGuardUpgradeable.__gap (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/security/ReentrancyGuardUpgradeable.sol#74) is never used in Offersx (jobs/bracket/brackets/brownie/contracts/Offersx.sol#15-94)
Structlibx.ETH (jobs/bracket/brackets/brownie/contracts/Structlibx.sol#13) is never used in Structlibx (jobs/bracket/brackets/brownie/contracts/Structlibx.sol#5-56)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variable

renounceOwnership() should be declared external:
        - OwnableUpgradeable.renounceOwnership() (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/access/OwnableUpgradeable.sol#66-68)
transferOwnership(address) should be declared external:
        - OwnableUpgradeable.transferOwnership(address) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/access/OwnableUpgradeable.sol#74-77)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
./jobs/bracket/brackets/brownie/contracts/Offersx.sol analyzed (11 contracts with 78 detectors), 89 result(s) found
```

Offersx.sol

```
MathUpgradeable.mulDiv(uint256,uint256,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#55-135) performs a multiplication on the result of a division:
        -denominator = denominator / twos (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#102)
        -inverse = (3 * denominator) ^ 2 (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#117)
MathUpgradeable.mulDiv(uint256,uint256,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#55-135) performs a multiplication on the result of a division:
        -denominator = denominator / twos (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#102)
        -inverse *= 2 - denominator * inverse (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#121)
MathUpgradeable.mulDiv(uint256,uint256,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#55-135) performs a multiplication on the result of a division:
        -denominator = denominator / twos (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#102)
        -inverse *= 2 - denominator * inverse (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#122)
MathUpgradeable.mulDiv(uint256,uint256,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#55-135) performs a multiplication on the result of a division:
        -denominator = denominator / twos (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#102)
        -inverse *= 2 - denominator * inverse (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#123)
MathUpgradeable.mulDiv(uint256,uint256,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#55-135) performs a multiplication on the result of a division:
        -denominator = denominator / twos (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#102)
        -inverse *= 2 - denominator * inverse (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#124)
MathUpgradeable.mulDiv(uint256,uint256,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#55-135) performs a multiplication on the result of a division:
        -denominator = denominator / twos (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#102)
        -inverse *= 2 - denominator * inverse (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#125)
MathUpgradeable.mulDiv(uint256,uint256,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#55-135) performs a multiplication on the result of a division:
        -denominator = denominator / twos (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#102)
        -inverse *= 2 - denominator * inverse (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#126)
MathUpgradeable.mulDiv(uint256,uint256,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#55-135) performs a multiplication on the result of a division:
        -prod0 = prod0 / twos (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#105)
        -result = prod0 * inverse (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#132)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply

PricingSequencer.checkSequencerState() (jobs/bracket/brackets/brownie/contracts/PricingSequencer.sol#67-76) uses timestamp for comparisons
        Dangerous comparisons:
        - answer == 0 && (block.timestamp - updatedAt) < 3600 (jobs/bracket/brackets/brownie/contracts/PricingSequencer.sol#72)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp

AddressUpgradeable.verifyCallResult(bool,bytes,string) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/AddressUpgradeable.sol#174-194) uses assembly
        - INLINE ASM (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/AddressUpgradeable.sol#186-189)
MathUpgradeable.mulDiv(uint256,uint256,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#55-135) uses assembly
        - INLINE ASM (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#66-70)
        - INLINE ASM (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#86-93)
        - INLINE ASM (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#100-109)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

Different versions of Solidity are used:
        - Version used: ['^0.8.0', '^0.8.1', '^0.8.2', '^0.8.4']
        - ^0.8.2 (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/proxy/utils/Initializable.sol#4)
        - ^0.8.1 (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/AddressUpgradeable.sol#4)
        - ^0.8.0 (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#4)
        - ^0.8.0 (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/SafeMathUpgradeable.sol#4)
        - ^0.8.0 (.brownie/packages/smartcontractkit/chainlink-brownie-contracts@0.3.1/contracts/src/v0.8/interfaces/AggregatorInterface.sol#2)
        - ^0.8.0 (.brownie/packages/smartcontractkit/chainlink-brownie-contracts@0.3.1/contracts/src/v0.8/interfaces/AggregatorV2V3Interface.sol#2)
        - ^0.8.0 (.brownie/packages/smartcontractkit/chainlink-brownie-contracts@0.3.1/contracts/src/v0.8/interfaces/AggregatorV3Interface.sol#2)
        - ^0.8.4 (jobs/bracket/brackets/brownie/contracts/IPricing.sol#3)
        - ^0.8.4 (jobs/bracket/brackets/brownie/contracts/PricingSequencer.sol#2)
        - ^0.8.4 (jobs/bracket/brackets/brownie/contracts/Structlibx.sol#3)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used
```

AUTOMATED TESTING

```
AddressUpgradeable.functionCall(address,bytes) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/AddressUpgradeable.sol#85-87) is never used and should be removed
AddressUpgradeable.functionCall(address,bytes,string) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/AddressUpgradeable.sol#95-101) is never used and should be removed
AddressUpgradeable.functionCallWithValue(address,bytes,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/AddressUpgradeable.sol#114-120) is never used and should be removed
AddressUpgradeable.functionCallWithValue(address,bytes,uint256,string) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/AddressUpgradeable.sol#128-139) is never used and should be removed
AddressUpgradeable.functionStaticCall(address,bytes) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/AddressUpgradeable.sol#147-149) is never used and should be removed
AddressUpgradeable.functionStaticCall(address,bytes,string) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/AddressUpgradeable.sol#157-166) is never used and should be removed
AddressUpgradeable.sendValue(address,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/AddressUpgradeable.sol#60-65) is never used and should be removed
AddressUpgradeable.verifyCallResult(bool,bytes,string) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/AddressUpgradeable.sol#174-194) is never used and should be removed
Initializable._disableInitializers() (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/proxy/utils/Initializable.sol#131-137) is never used and should be removed
MathUpgradeable.average(uint256,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#34-37) is never used and should be removed
MathUpgradeable.ceilDiv(uint256,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#45-48) is never used and should be removed
MathUpgradeable.max(uint256,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#19-21) is never used and should be removed
MathUpgradeable.min(uint256,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#26-28) is never used and should be removed
MathUpgradeable.mulDiv(uint256,uint256,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#55-135) is never used and should be removed
MathUpgradeable.mulDiv(uint256,uint256,uint256,MathUpgradeable.Rounding) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#140-151) is never used and should be removed
MathUpgradeable.sqrt(uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#158-214) is never used and should be removed
MathUpgradeable.sqrt(uint256,MathUpgradeable.Rounding) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#219-225) is never used and should be removed
SafeMathUpgradeable.add(uint256,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/SafeMathUpgradeable.sol#93-95) is never used and should be removed
SafeMathUpgradeable.div(uint256,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/SafeMathUpgradeable.sol#135-137) is never used and should be removed
SafeMathUpgradeable.div(uint256,uint256,string) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/SafeMathUpgradeable.sol#191-200) is never used and should be removed
SafeMathUpgradeable.mod(uint256,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/SafeMathUpgradeable.sol#151-153) is never used and should be removed
SafeMathUpgradeable.mod(uint256,uint256,string) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/SafeMathUpgradeable.sol#217-226) is never used and should be removed
SafeMathUpgradeable.mul(uint256,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/SafeMathUpgradeable.sol#123-125) is never used and should be removed
SafeMathUpgradeable.sub(uint256,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/SafeMathUpgradeable.sol#107-109) is never used and should be removed
SafeMathUpgradeable.sub(uint256,uint256,string) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/SafeMathUpgradeable.sol#168-177) is never used and should be removed
SafeMathUpgradeable.tryAdd(uint256,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/SafeMathUpgradeable.sol#22-28) is never used and should be removed
SafeMathUpgradeable.trySub(uint256,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/SafeMathUpgradeable.sol#54-60) is never used and should be removed
SafeMathUpgradeable.tryMul(uint256,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/SafeMathUpgradeable.sol#76-81) is never used and should be removed
SafeMathUpgradeable.tryDiv(uint256,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/SafeMathUpgradeable.sol#47-57) is never used and should be removed
SafeMathUpgradeable.trySub(uint256,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/SafeMathUpgradeable.sol#35-40) is never used and should be removed
StructLibx.mulFactor(uint256,uint256) (jobs/bracket/brackets/brownie/contracts/StructLibx.sol#49-55) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.8.2 (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/proxy/utils/Initializable.sol#4) allows old versions
Pragma version^0.8.1 (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/AddressUpgradeable.sol#4) allows old versions
Pragma version^0.8.0 (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#4) allows old versions
Pragma version^0.8.0 (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/SafeMathUpgradeable.sol#4) allows old versions
Pragma version^0.8.0 (.brownie/packages/smartcontractkit/chainlink-brownie-contracts@0.3.1/contracts/src/v0.8/interfaces/AggregatorInterface.sol#2) allows old versions
Pragma version^0.8.0 (.brownie/packages/smartcontractkit/chainlink-brownie-contracts@0.3.1/contracts/src/v0.8/interfaces/AggregatorV2V3Interface.sol#2) allows old versions
Pragma version^0.8.0 (.brownie/packages/smartcontractkit/chainlink-brownie-contracts@0.3.1/contracts/src/v0.8/interfaces/AggregatorV3Interface.sol#2) allows old versions
solc-0.8.16 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in AddressUpgradeable.sendValue(address,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/AddressUpgradeable.sol#60-65):
        - (success) = recipient.call{value: amount}() (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/AddressUpgradeable.sol#63)
Low level call in AddressUpgradeable.functionCallWithValue(address,bytes,uint256,string) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/AddressUpgradeable.sol#128-139):
        - (success,returndata) = target.call{value: value}(data) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/AddressUpgradeable.sol#137)
Low level call in AddressUpgradeable.functionStaticCall(address,bytes,string) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/AddressUpgradeable.sol#157-166):
        - (success,returndata) = target.staticcall(data) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/AddressUpgradeable.sol#164)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

StructLibx.ETH (jobs/bracket/brackets/brownie/contracts/StructLibx.sol#13) is never used in StructLibx (jobs/bracket/brackets/brownie/contracts/StructLibx.sol#8-56)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variable
./jobs/bracket/brackets/brownie/contracts/PricingSequencer.sol analyzed (10 contracts with 78 detectors), 54 result(s) found
```

StructLibx.sol

```
MathUpgradeable.mulDiv(uint256,uint256,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#55-135) performs a multiplication on the result of a division:
        -denominator = denominator / twos (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#102)
        -inverse = (3 * denominator) ^ 2 (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#117)
MathUpgradeable.mulDiv(uint256,uint256,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#55-135) performs a multiplication on the result of a division:
        -denominator = denominator / twos (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#102)
        -inverse *= 2 - denominator * inverse (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#121)
MathUpgradeable.mulDiv(uint256,uint256,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#55-135) performs a multiplication on the result of a division:
        -denominator = denominator / twos (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#102)
        -inverse *= 2 - denominator * inverse (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#122)
MathUpgradeable.mulDiv(uint256,uint256,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#55-135) performs a multiplication on the result of a division:
        -denominator = denominator / twos (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#102)
        -inverse *= 2 - denominator * inverse (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#123)
MathUpgradeable.mulDiv(uint256,uint256,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#55-135) performs a multiplication on the result of a division:
        -denominator = denominator / twos (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#102)
        -inverse *= 2 - denominator * inverse (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#124)
MathUpgradeable.mulDiv(uint256,uint256,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#55-135) performs a multiplication on the result of a division:
        -denominator = denominator / twos (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#102)
        -inverse *= 2 - denominator * inverse (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#125)
MathUpgradeable.mulDiv(uint256,uint256,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#55-135) performs a multiplication on the result of a division:
        -denominator = denominator / twos (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#102)
        -inverse *= 2 - denominator * inverse (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#126)
MathUpgradeable.mulDiv(uint256,uint256,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#55-135) performs a multiplication on the result of a division:
        -prod0 = prod0 / twos (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#105)
        -result = prod0 * inverse (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#132)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply

MathUpgradeable.mulDiv(uint256,uint256,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#55-135) uses assembly
        - INLINE ASM (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#66-70)
        - INLINE ASM (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#86-93)
        - INLINE ASM (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#100-109)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

Different versions of Solidity are used:
        - Version used: ['^0.8.0', '^0.8.4']
        - ^0.8.0 (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#4)
        - ^0.8.0 (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/SafeMathUpgradeable.sol#4)
        - ^0.8.4 (jobs/bracket/brackets/brownie/contracts/StructLibx.sol#3)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used

MathUpgradeable.average(uint256,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#34-37) is never used and should be removed
MathUpgradeable.ceilDiv(uint256,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#45-48) is never used and should be removed
MathUpgradeable.max(uint256,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#19-21) is never used and should be removed
MathUpgradeable.min(uint256,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#26-28) is never used and should be removed
MathUpgradeable.mulDiv(uint256,uint256,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#55-135) is never used and should be removed
MathUpgradeable.mulDiv(uint256,uint256,uint256,MathUpgradeable.Rounding) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#140-151) is never used and should be removed
MathUpgradeable.sqrt(uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#158-214) is never used and should be removed
MathUpgradeable.sqrt(uint256,MathUpgradeable.Rounding) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#219-225) is never used and should be removed
SafeMathUpgradeable.add(uint256,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/SafeMathUpgradeable.sol#93-95) is never used and should be removed
SafeMathUpgradeable.div(uint256,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/SafeMathUpgradeable.sol#135-137) is never used and should be removed
SafeMathUpgradeable.div(uint256,uint256,string) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/SafeMathUpgradeable.sol#191-200) is never used and should be removed
SafeMathUpgradeable.mod(uint256,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/SafeMathUpgradeable.sol#151-153) is never used and should be removed
SafeMathUpgradeable.mod(uint256,uint256,string) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/SafeMathUpgradeable.sol#217-226) is never used and should be removed
SafeMathUpgradeable.mul(uint256,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/SafeMathUpgradeable.sol#123-125) is never used and should be removed
SafeMathUpgradeable.sub(uint256,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/SafeMathUpgradeable.sol#107-109) is never used and should be removed
SafeMathUpgradeable.sub(uint256,uint256,string) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/SafeMathUpgradeable.sol#168-177) is never used and should be removed
SafeMathUpgradeable.tryAdd(uint256,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/SafeMathUpgradeable.sol#22-28) is never used and should be removed
SafeMathUpgradeable.trySub(uint256,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/SafeMathUpgradeable.sol#54-60) is never used and should be removed
SafeMathUpgradeable.tryMul(uint256,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/SafeMathUpgradeable.sol#76-81) is never used and should be removed
SafeMathUpgradeable.tryDiv(uint256,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/SafeMathUpgradeable.sol#47-57) is never used and should be removed
SafeMathUpgradeable.trySub(uint256,uint256) (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/SafeMathUpgradeable.sol#35-40) is never used and should be removed
StructLibx.mulFactor(uint256,uint256) (jobs/bracket/brackets/brownie/contracts/StructLibx.sol#49-55) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.8.0 (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/MathUpgradeable.sol#4) allows old versions
Pragma version^0.8.0 (.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.7.0/contracts-upgradeable/utils/math/SafeMathUpgradeable.sol#4) allows old versions
solc-0.8.16 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

StructLibx.ETH (jobs/bracket/brackets/brownie/contracts/StructLibx.sol#13) is never used in StructLibx (jobs/bracket/brackets/brownie/contracts/StructLibx.sol#8-56)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variable
./jobs/bracket/brackets/brownie/contracts/StructLibx.sol analyzed (3 contracts with 78 detectors), 36 result(s) found
```

- No major issues found by Slither.

THANK YOU FOR CHOOSING

// HALBORN