// HALBORN

HighStreetMarket - ProductToken

Smart Contract Security Audit

Prepared by: **Halborn**

Date of Engagement: **December 27th, 2021 - December 31st, 2021**

Visit: **Halborn.com**

DOCUMENT REVISION HISTORY

VERSION	MODIFICATION	DATE	AUTHOR
0.1	Document Creation	12/27/2021	Roberto Reigada
0.2	Document Updates	12/31/2021	Roberto Reigada
0.3	Draft Review	01/03/2022	Gabi Urrutia
1.0	Remediation Plan	01/10/2021	Roberto Reigada
1.1	Remediation Plan Review	01/10/2021	Gabi Urrutia

CONTACTS

CONTACT	COMPANY	EMAIL
Rob Behnke	Halborn	Rob.Behnke@halborn.com
Steven Walbroehl	Halborn	Steven.Walbroehl@halborn.com
Gabi Urrutia	Halborn	Gabi.Urrutia@halborn.com
Roberto Reigada	Halborn	Roberto.Reigada@halborn.com

EXECUTIVE OVERVIEW

1.1 INTRODUCTION

HighStreetMarket engaged Halborn to conduct a security audit on their smart contracts beginning on December 27th, 2021 and ending on December 31st, 2021. The security assessment was scoped to the smart contracts provided in the Github repository Highstreet-World/ProductToken

1.2 AUDIT SUMMARY

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

The purpose of this audit is to:

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

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

1.3 TEST APPROACH & METHODOLOGY

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

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

RISK METHODOLOGY:

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

RISK SCALE - LIKELIHOOD

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

RISK SCALE - IMPACT

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

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

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

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL

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

1.4 SCOPE

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

- BancorBondingCurve.sol
- Power.sol
- ProductNft.sol
- ProductTokenCore.sol
- ProductTokenHighBase.sol

Commit ID: 9be0879308ad3e4fa43a8793eb5e580f60e291b1
Fixed Commit ID:

OUT-OF-SCOPE:
Other smart contracts in the repository, external libraries and economical attacks.

2. ASSESSMENT SUMMARY & FINDINGS OVERVIEW

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL
0	1	0	2	2

LIKELIHOOD

IMPACT

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

EXECUTIVE OVERVIEW

SECURITY ANALYSIS	RISK LEVEL	REMEDIATION DATE
HAL-01 - PRODUCTTOKENHIGHBASE CONTRACT IS VULNERABLE TO SANDWICH ATTACKS	High	SOLVED - 01/05/2022
HAL-02 - MISSING ZERO ADDRESS CHECK	Low	SOLVED - 01/05/2022
HAL-03 - EXTERNAL CALLS WITHIN A LOOP	Low	SOLVED - 01/05/2022
HAL-04 - POSSIBLE MISUSE OF PUBLIC FUNCTIONS	Informational	SOLVED - 01/05/2022
HAL-05 - TYPO IN FUNCTION NAME	Informational	SOLVED - 01/05/2022

FINDINGS & TECH DETAILS

3.1 (HAL-01) PRODUCTTOKENHIGHBASE CONTRACT IS VULNERABLE TO SANDWICH ATTACKS - HIGH

Description:

The contract ProductTokenHighBase is an ERC20 contract that allows users to buy and sell a product token. The buying and selling price of the token is based on a pricing algorithm called Bonding Curve. This algorithm causes the price of the token to increase with every purchase, meaning that early buyers are rewarded as they can, then, sell the tokens back for a profit.

The function buy(uint256 maxPrice_) allows users to buy a product token with a maximum acceptable price. It only allows buying 1 token per call:

Listing 1: ProductTokenHighBase.sol

```
78 function buy(uint256 maxPrice_) external virtual whenNotPaused
       nonReentrant {
79   if(endTime[FEATURE_ENDTIME_BUY] != 0) {
80     require(now256() < endTime[FEATURE_ENDTIME_BUY], "sale is
           expire");
81   }
82   require(maxPrice_ > 0, "invalid max price");
83
84   transferFromHighToken(_msgSender(), address(this), maxPrice_);
85
86   (uint256 change)  = _buy(maxPrice_);
87   if(change > 0) {
88     transferHighToken(_msgSender(), change);
89   }
90 }
```

The function sell(uint32 amount_) allows users to sell a product token, returning the equivalent HIGH tokens:

Listing 2: ProductTokenHighBase.sol

```
100 function sell(uint32 amount_) external virtual whenNotPaused
      nonReentrant {
101   if(endTime[FEATURE_ENDTIME_SELL] != 0) {
102     require(now256() < endTime[FEATURE_ENDTIME_SELL], "sale is
          expire");
103   }
104   require(amount_ > 0, "Amount must be non-zero.");
105   require(balanceOf(_msgSender()) >= amount_, "Insufficient tokens
        .");
106
107   uint256 price = _sell(amount_);
108   transferHighToken(_msgSender(), price);
109 }
```

The growth of the price with each purchase is determined by the different bonding curve parameters, which are set in the initialize() function of the ProductTokenHighBase contract:

Listing 3: ProductTokenHighBase.sol (Lines 47-50)

```
27 /**
28  * @dev initializer function.
29  *
30  * @param _name the name of this token
31  * @param _symbol the symbol of this token
32  * @param _bondingCurve bonding curve instance address
33  * @param _productNft product nft instance address
34  * @param _reserveRatio the reserve ratio in the curve function.
      Number in parts per million
35  * @param _maxTokenCount the amount of token that will exist for
      this type.
36  * @param _supplyOffset this amount is used to determine initial
      price.
37  * @param _baseReserve the base amount of reserve tokens, in
      accordance to _supplyOffset.
38  * @param _endTime end time is the last time when user can buy and
       sell;
39  *
40 */
41 function initialize(
42   string memory _name,
```

```
43    string memory _symbol,
44    address _high,
45    address _bondingCurve,
46    address _productNft,
47    uint32 _reserveRatio,
48    uint32 _maxTokenCount,
49    uint32 _supplyOffset,
50    uint256 _baseReserve,
51    uint256 _endTime
52  ) public virtual initializer{
53    HIGH = _high;
54    ProductTokenCore.initialize(
55      _name,
56      _symbol,
57      _bondingCurve,
58      _productNft,
59      _reserveRatio,
60      _maxTokenCount,
61      _supplyOffset,
62      _baseReserve
63    );
64    if(_endTime > 0) {
65      updateEndTime(FEATURE_ENDTIME_MAX, _endTime);
66    }
67  }
```

As there are no restriction or cooldown period to sell the product token after its purchase, the buy() function calls are vulnerable to a sandwich attack.

Proof of Concept:

Based in the parameters found in the 2_deploy_contract.js file:
- _reserveRatio the reserve ratio in the curve function. Number in parts per million -> 30000
- _maxTokenCount the amount of token that will exist for this type. -> 250
- _supplyOffset this amount is used to determine initial price. -> 700
- _baseReserve the base amount of reserve tokens. -> 800

1. Attacker scans the mempool for 2 ProductTokenHighBase.buy() transactions.
2. Attacker detects 2 ProductTokenHighBase.buy() txs in the mempool coming from user1 and user2.
3. Attacker frontruns both txs and calls himself ProductTokenHighBase .buy().
4. If both users had set the maxPrice_ in ProductTokenHighBase.buy() to the exact price (ProductTokenHighBase.getCurrentPrice()), their txs would revert. If that was the case, it makes sense to think that they would resubmit their transactions with a higher maxPrice_. On the other hand, if both users had set the maxPrice_ in ProductTokenHighBase.buy() with some margin, their txs would be completed but they would be paying a higher price for the ProductTokenHighBase product token.
5. Attacker backruns both txs (user1 and user2 txs) and sells the product token bought initially, getting a 3.33% profit:

Tokens paid initially by the attacker (during the frontrun): 40_547860657636861605
Tokens paid by user1: 42_461906982198102077
Tokens paid by user2: 44_463383637847924478
Total tokens received by the attacker after the sale of the product token (during the backrun): 41_898188427972082682

Profitted tokens after sandwiching 2 users = 1_350327770335221077
% of profit: (1_350327770335221077 / 40_547860657636861605)x 100 = 3.33%

With these bonding curve parameters, two additional purchases are needed before selling to make profit. With different bonding curve parameters, it would be possible to profit by just sandwiching one transaction instead of two, increasing this attack likelihood.

Risk Level:

Likelihood - 3

Impact - 5

Recommendation:

It is recommended to set the bonding curve parameters of the ProductTokenHighBase contract in order that, before a user can profit by selling the product token, a number high enough of buy orders are needed.

On the other hand, perhaps a simpler approach, another possible solution would be to add a cooldown on the sell() function, not allowing users to sell right away after a buy() call.

Remediation Plan:

SOLVED: The HighStreetMarket team solved the issue in the following commit IDs:
- 3d83211d1720c6cd1ae1cd21631a400c4932bad7
- 27d266184835bdf54aa3c8135a7d2a2a89fc7b37

A cooldown has been added that restricts users from calling sell(), transferFrom() and transfer() functions for a fixed period of time.

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

Description:

The contract ProductTokenHighBase is missing zero address checks in its initializate() function. Every address should be validated and checked that is different from zero.

Code location:

Listing 4: ProductTokenHighBase.sol

```
41 function initialize(
42   string memory _name,
43   string memory _symbol,
44   address _high,
45   address _bondingCurve,
46   address _productNft,
47   uint32 _reserveRatio,
48   uint32 _maxTokenCount,
49   uint32 _supplyOffset,
50   uint256 _baseReserve,
51   uint256 _endTime
52 ) public virtual initializer{
53   HIGH = _high;
54   ProductTokenCore.initialize(
55     _name,
56     _symbol,
57     _bondingCurve,
58     _productNft,
59     _reserveRatio,
60     _maxTokenCount,
61     _supplyOffset,
62     _baseReserve
63   );
64   if(_endTime > 0) {
65     updateEndTime(FEATURE_ENDTIME_MAX, _endTime);
66   }
67 }
```

Risk Level:

Likelihood - 3
Impact - 2

Recommendation:

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

Remediation Plan:

SOLVED: The HighStreetMarket team solved the issue in the commit ID 84746701f0463a77d75c0ec8d7444784e594e42d

3.3 (HAL-03) EXTERNAL CALLS WITHIN A LOOP - LOW

Description:

In the contract ProductTokenHighBase the function tradein() allows users to trade in their Product token for a NFT.

Listing 5: ProductTokenHighBase.sol (Lines 122)

```
116 function tradein(uint32 amount_) external virtual whenNotPaused
        nonReentrant{
117   if(endTime[FEATURE_ENDTIME_TRADEIN] != 0) {
118     require(now256() < endTime[FEATURE_ENDTIME_TRADEIN], "sale is
          expire");
119   }
120   require(amount_ > 0, "Amount must be non-zero.");
121   require(balanceOf(_msgSender()) >= amount_, "Insufficient tokens
        .");
122   _tradein(amount_);
123 }
```

Listing 6: ProductTokenCore.sol (Lines 466)

```
453 function _tradein(uint32 amount_) internal virtual {
454
455   (, uint256 fee) = _sellReturn(amount_);
456
457   _burn(_msgSender(), amount_);
458   // redeem value should give to supplier
459   uint256 tradinReturn = _tradinReturn(amount_);
460
461   // 50% of fee and redeem price are for supplier
462   _updateSupplierFee(fee * FEE_MULTIPLIER / FEE_RATE_IN_SELL +
        tradinReturn);
463   // 50% of fee, is for plateform
464   _updatePlateformFee(fee * (FEE_RATE_IN_SELL - FEE_MULTIPLIER) /
        FEE_RATE_IN_SELL);
465
466   nft.mintBatch(_msgSender(), tradeinCount, amount_);
467
```

```
468   tradeinCount = tradeinCount + amount_;
469   tradeinReserveBalance = tradeinReserveBalance + tradinReturn;
470
471   emit Tradein(_msgSender(), amount_, tradinReturn, fee);
472 }
```

If a user trades in 10 Product tokens, 10 NFTs will be minted through the function ProductNft.mintBatch():

Listing 7: ProductNft.sol (Lines 145)

```
140   function mintBatch(address to_, uint256 start_, uint256 count_)
          external {
141     require(minters[_msgSender()] == true, 'permission denied');
142     require(to_ != address(0), 'invalid receiver');
143     require(start_ + count_ < maxCount, "cap exceeded");
144
145     for(uint256 index = start_; index < start_ + count_; index ++)
            {
146         _safeMint(to_, index);
147     }
148     emit MintBatch(to_, start_, count_);
149   }
```

The gas cost of minting 10 NFTs is 1.258.049. This would already exceed, for example, Metamask default gas limit of 1,200,000 GWEI. Users not aware of this may submit the transaction to then find out that the transaction have failed with an Out of gas message. A transaction of 1,200,000 GWEI in gas is worth $451.95 (date: 01/01/2022). This is the potential loss that can happen to a user not aware of this.

For that reason, it is recommended to limit the amount_ parameter of the tradein() function.

Proof of Concept:

```
Calling -> contract_ProductTokenHighBase.tradein(10, {'from': user1})
Transaction sent: 0x0ef9e8a0c66c7424fc520a2f6e6228d4d83f7128992b1f86198c431a736e0840
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 21
  ProductTokenHighBase.tradein confirmed   Block: 13921317   Gas used: 1258049 (18.72%)

Call trace for '0x0ef9e8a0c66c7424fc520a2f6e6228d4d83f7128992b1f86198c431a736e0840':
Initial call cost  [-12996 gas]
ProductTokenHighBase.tradein  0:43409  [30062 / 1181045 gas]
├── ERC20Upgradeable.balanceOf  251:19551  [3214 / 78733 gas]
│   ├── ProductTokenCore._tradein  281:403  [38 / 2798 gas]
│   │   └── ProductTokenCore._sellReturn  287:398  [960 / 2760 gas]
│   │       └── ProductTokenCore._getTotalSupply  304:366  [1800 gas]
│   │
│   └── BancorBondingCurve.calculateSaleReturn  [STATICCALL]  479:19422  [1715 / 72721 gas]
│       ├── address: contract_BancorBondingCurve.address
│       ├── input arguments:
│       │   ├── _supply: 710
│       │   ├── _reserveBalance: 128361153298059138l821
│       │   ├── _reserveRatio: 30000
│       │   └── _sellAmount: 10
│       └── return value: 483611532980591381825
│
│       ├── SafeMathUpgradeable.sub  709:736  [91 gas]
│       ├── Power.power  746:19271  [394 / 70624 gas]
│       │   ├── Power.ln  761:15535  [51424 gas]
│       │   ├── Power.findPositionInMaxExpArray  15585:16541  [9596 gas]
│       │   └── Power.fixedExp  16582:19258  [9210 gas]
│       ├── SafeMathUpgradeable.mul  19283:19317  [116 gas]
│       ├── SafeMathUpgradeable.sub  19333:19360  [91 gas]
│       └── SafeMathUpgradeable.div  19364:19388  [84 gas]
├── ERC20Upgradeable._burn  19566:19695  [-16173 gas]
├── ProductTokenCore._tradinReturn  19701:19764  [1003 gas]
│
├── BancorBondingCurve.calculatePriceForNTokens  [STATICCALL]  19882:38215  [1735 / 70742 gas]
│   ├── address: contract_BancorBondingCurve.address
│   ├── input arguments:
│   │   ├── _supply: 710
│   │   ├── _reserveBalance: 800000000000000000000
│   │   ├── _reserveRatio: 30000
│   │   └── _amount: 10
│   └── return value: 475150887200091539313
│
│   ├── SafeMathUpgradeable.add  20107:20135  [94 gas]
│   ├── Power.power  20145:38112  [394 / 68797 gas]
│   │   ├── Power.ln  20160:34376  [49597 gas]
│   │   ├── Power.findPositionInMaxExpArray  34426:35382  [9596 gas]
│   │   └── Power.fixedExp  35423:38099  [9210 gas]
│   └── SafeMathUpgradeable.mul  38127:38161  [116 gas]
├── ProductTokenCore._updateSupplierFee  38344:38372  [5890 gas]
├── ProductTokenCore._updatePlateformFee  38440:38468  [5890 gas]
│
└── ProductNft.mintBatch  [CALL]  38549:43249  [1004898 gas]
    ├── address: contract_ProductNft.address
    ├── value: 0
    └── input arguments:
        ├── to_: user1.address
        ├── start_: 0
        └── count_: 10
```

Risk Level:

Likelihood - 2

Impact - 2

Recommendation:

It is recommended to limit the amount_ parameter of the tradein() function, making sure that is, for example, lower than 10.

Remediation Plan:

SOLVED: The HighStreetMarket team solved the issue in the commit ID 56f794ef686c2840afd35f7255ce8cb23bba8127

3.4 (HAL-04) POSSIBLE MISUSE OF PUBLIC FUNCTIONS - INFORMATIONAL

Description:

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

BancorBondingCurve.sol
- calculatePriceForNTokens() (BancorBondingCurve.sol#38-64)
- calculatePurchaseReturn() (BancorBondingCurve.sol#81-109)
- calculateSaleReturn() (BancorBondingCurve.sol#125-159)

ProductTokenCore.sol
- getCurrentPrice() (ProductTokenCore.sol#260-263)
- calculateBuyReturn() (ProductTokenCore.sol#329-334)
- calculateSellReturn() (ProductTokenCore.sol#363-367)

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

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

Remediation Plan:

SOLVED: The HighStreetMarket team solved the issue in the commit ID 674c5f7a814acf33bafaedd43c94df0731e69c73

3.5 (HAL-05) TYPO IN FUNCTION NAMES AND EVENTS - INFORMATIONAL

Description:

The following functions, variables, and events in the ProductTokenCore contract have a typo: Plateform instead of Platform.

- Line 64: `uint256 private plateformFee;`
- Line 150: `event ClaimPlateformFee(address indexed sender, uint256 amount);`
- Line 411: `_updatePlateformFee(fee * (FEE_RATE_IN_BUY - FEE_MULTIPLIER)/ FEE_RATE_IN_BUY);`
- Line 440: `_updatePlateformFee(fee * (FEE_RATE_IN_SELL - FEE_MULTIPLIER)/ FEE_RATE_IN_SELL);`
- Line 464: `_updatePlateformFee(fee * (FEE_RATE_IN_SELL - FEE_MULTIPLIER)/ FEE_RATE_IN_SELL);`
- Line 488: `function _updatePlateformFee(uint256 fee_)internal virtual {`
- Line 489: `plateformFee = plateformFee + fee_;`
- Line 551: `function getPlateformFee()external view virtual returns( uint256 amount){`
- Line 553: `return plateformFee;`
- Line 562: `function claimPlateformFee(uint256 amount_)external virtual {`
- Line 564: `require(amount_ <= plateformFee, "amount is exceed");`
- Line 567: `plateformFee = plateformFee - amount_;`
- Line 569: `emit ClaimPlateformFee(_msgSender(), amount_);`

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

It is recommended to correct the typo in all the functions, variables, and events mentioned.

Remediation Plan:

SOLVED: The HighStreetMarket team solved the issue in the commit ID 97746779dd28d751e1d1ce87b2c123300928704c

AUTOMATED TESTING

4.1 STATIC ANALYSIS REPORT

Description:

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

Slither results:

BancorBondingCurve.sol

```
Power.ln(uint256,uint256) (contracts/Power.sol#205-230) performs a multiplication on the result of a division:
        -x = _numerator * FIXED_1 / _denominator (contracts/Power.sol#209)
        -x = (x * x) / FIXED_1 (contracts/Power.sol#221)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply

Different versions of Solidity is used:
        - Version used: ['^0.8.0', '^0.8.2']
        - ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/proxy/utils/Initializable.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/utils/math/SafeMathUpgradeable.sol#4)
        - ^0.8.2 (contracts/BancorBondingCurve.sol#3)
        - ^0.8.2 (contracts/Power.sol#3)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used

SafeMathUpgradeable.div(uint256,uint256,string) (node_modules/@openzeppelin/contracts-upgradeable/utils/math/SafeMathUpgradeable.sol#191-200) is never used and should be removed
SafeMathUpgradeable.mod(uint256,uint256) (node_modules/@openzeppelin/contracts-upgradeable/utils/math/SafeMathUpgradeable.sol#151-153) is never used and should be removed
SafeMathUpgradeable.mod(uint256,uint256,string) (node_modules/@openzeppelin/contracts-upgradeable/utils/math/SafeMathUpgradeable.sol#217-226) is never used and should be removed
SafeMathUpgradeable.sub(uint256,uint256,string) (node_modules/@openzeppelin/contracts-upgradeable/utils/math/SafeMathUpgradeable.sol#168-177) is never used and should be removed
SafeMathUpgradeable.tryAdd(uint256,uint256) (node_modules/@openzeppelin/contracts-upgradeable/utils/math/SafeMathUpgradeable.sol#22-28) is never used and should be removed
SafeMathUpgradeable.tryDiv(uint256,uint256) (node_modules/@openzeppelin/contracts-upgradeable/utils/math/SafeMathUpgradeable.sol#64-69) is never used and should be removed
SafeMathUpgradeable.tryMod(uint256,uint256) (node_modules/@openzeppelin/contracts-upgradeable/utils/math/SafeMathUpgradeable.sol#76-81) is never used and should be removed
SafeMathUpgradeable.tryMul(uint256,uint256) (node_modules/@openzeppelin/contracts-upgradeable/utils/math/SafeMathUpgradeable.sol#47-57) is never used and should be removed
SafeMathUpgradeable.trySub(uint256,uint256) (node_modules/@openzeppelin/contracts-upgradeable/utils/math/SafeMathUpgradeable.sol#35-40) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/proxy/utils/Initializable.sol#4) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/utils/math/SafeMathUpgradeable.sol#4) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.2 (contracts/BancorBondingCurve.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.2 (contracts/Power.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
solc-0.8.9 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Parameter BancorBondingCurve.calculatePriceForNTokens(uint32,uint256,uint32,uint32)._supply (contracts/BancorBondingCurve.sol#39) is not in mixedCase
Parameter BancorBondingCurve.calculatePriceForNTokens(uint32,uint256,uint32,uint32)._reserveBalance (contracts/BancorBondingCurve.sol#40) is not in mixedCase
Parameter BancorBondingCurve.calculatePriceForNTokens(uint32,uint256,uint32,uint32)._reserveRatio (contracts/BancorBondingCurve.sol#41) is not in mixedCase
Parameter BancorBondingCurve.calculatePriceForNTokens(uint32,uint256,uint32,uint32)._amount (contracts/BancorBondingCurve.sol#42) is not in mixedCase
Parameter BancorBondingCurve.calculatePurchaseReturn(uint32,uint256,uint32,uint256)._supply (contracts/BancorBondingCurve.sol#82) is not in mixedCase
Parameter BancorBondingCurve.calculatePurchaseReturn(uint32,uint256,uint32,uint256)._reserveBalance (contracts/BancorBondingCurve.sol#83) is not in mixedCase
Parameter BancorBondingCurve.calculatePurchaseReturn(uint32,uint256,uint32,uint256)._reserveRatio (contracts/BancorBondingCurve.sol#84) is not in mixedCase
Parameter BancorBondingCurve.calculatePurchaseReturn(uint32,uint256,uint32,uint256)._depositAmount (contracts/BancorBondingCurve.sol#85) is not in mixedCase
Parameter BancorBondingCurve.calculateSaleReturn(uint32,uint256,uint32,uint32)._supply (contracts/BancorBondingCurve.sol#126) is not in mixedCase
Parameter BancorBondingCurve.calculateSaleReturn(uint32,uint256,uint32,uint32)._reserveBalance (contracts/BancorBondingCurve.sol#127) is not in mixedCase
Parameter BancorBondingCurve.calculateSaleReturn(uint32,uint256,uint32,uint32)._reserveRatio (contracts/BancorBondingCurve.sol#128) is not in mixedCase
Parameter BancorBondingCurve.calculateSaleReturn(uint32,uint256,uint32,uint32)._sellAmount (contracts/BancorBondingCurve.sol#129) is not in mixedCase
Function Power.__Power_init() (contracts/Power.sol#43-174) is not in mixedCase
Parameter Power.power(uint256,uint256,uint32,uint32)._baseN (contracts/Power.sol#192) is not in mixedCase
Parameter Power.power(uint256,uint256,uint32,uint32)._baseD (contracts/Power.sol#192) is not in mixedCase
Parameter Power.power(uint256,uint256,uint32,uint32)._expN (contracts/Power.sol#192) is not in mixedCase
Parameter Power.power(uint256,uint256,uint32,uint32)._expD (contracts/Power.sol#192) is not in mixedCase
Parameter Power.ln(uint256,uint256)._numerator (contracts/Power.sol#205) is not in mixedCase
Parameter Power.ln(uint256,uint256)._denominator (contracts/Power.sol#205) is not in mixedCase
Parameter Power.floorLog2(uint256)._n (contracts/Power.sol#235) is not in mixedCase
Parameter Power.findPositionInMaxExpArray(uint256)._x (contracts/Power.sol#263) is not in mixedCase
Parameter Power.fixedExp(uint256,uint8)._x (contracts/Power.sol#291) is not in mixedCase
Parameter Power.fixedExp(uint256,uint8)._precision (contracts/Power.sol#291) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Power.fixedExp(uint256,uint8) (contracts/Power.sol#291-361) uses literals with too many digits:
        - res += xi * 0x03442c4e6074a82f1797f72ac0000000 (contracts/Power.sol#296)
Power.fixedExp(uint256,uint8) (contracts/Power.sol#291-361) uses literals with too many digits:
        - res += xi * 0x0116b96f757c380fb287fd0e40000000 (contracts/Power.sol#298)
Power.fixedExp(uint256,uint8) (contracts/Power.sol#291-361) uses literals with too many digits:
        - res += xi * 0x0045ae5bdd5f0e03eca1ff4390000000 (contracts/Power.sol#300)
Power.fixedExp(uint256,uint8) (contracts/Power.sol#291-361) uses literals with too many digits:

        - res += xi * 0x000defabf91302cd95b9ffda50000000 (contracts/Power.sol#302)
Power.fixedExp(uint256,uint8) (contracts/Power.sol#291-361) uses literals with too many digits:
        - res += xi * 0x0002529ca9332b22439efff9b0000000 (contracts/Power.sol#304)
Power.fixedExp(uint256,uint8) (contracts/Power.sol#291-361) uses literals with too many digits:
        - res += xi * 0x000054f1cf12bd04e516b6da88000000 (contracts/Power.sol#306)
Power.fixedExp(uint256,uint8) (contracts/Power.sol#291-361) uses literals with too many digits:
        - res += xi * 0x00000a9e39e257a09ca2d6db51000000 (contracts/Power.sol#308)
Power.fixedExp(uint256,uint8) (contracts/Power.sol#291-361) uses literals with too many digits:
        - res += xi * 0x0000012e066e7b839fa050c309000000 (contracts/Power.sol#310)
Power.fixedExp(uint256,uint8) (contracts/Power.sol#291-361) uses literals with too many digits:
        - res += xi * 0x0000001e33d7d926c329a1ad1a800000 (contracts/Power.sol#312)
Power.fixedExp(uint256,uint8) (contracts/Power.sol#291-361) uses literals with too many digits:
        - res += xi * 0x00000002bee513bdb4a6b19b5f800000 (contracts/Power.sol#314)
Power.fixedExp(uint256,uint8) (contracts/Power.sol#291-361) uses literals with too many digits:
        - res += xi * 0x000000003a9316fa79b88eccf2a00000 (contracts/Power.sol#316)
Power.fixedExp(uint256,uint8) (contracts/Power.sol#291-361) uses literals with too many digits:
        - res += xi * 0x00000000048177ebe1fa812375200000 (contracts/Power.sol#318)
Power.fixedExp(uint256,uint8) (contracts/Power.sol#291-361) uses literals with too many digits:
        - res += xi * 0x00000000005263fe90242dcbacf00000 (contracts/Power.sol#320)
Power.fixedExp(uint256,uint8) (contracts/Power.sol#291-361) uses literals with too many digits:
        - res += xi * 0x000000000057e22099c030d94100000 (contracts/Power.sol#322)
Power.fixedExp(uint256,uint8) (contracts/Power.sol#291-361) uses literals with too many digits:
        - res += xi * 0x0000000000057e22099c030d9410000 (contracts/Power.sol#324)
Power.fixedExp(uint256,uint8) (contracts/Power.sol#291-361) uses literals with too many digits:
        - res += xi * 0x00000000000052b6b54569976310000 (contracts/Power.sol#326)
```

AUTOMATED TESTING

```
Power.fixedExp(uint256,uint8) (contracts/Power.sol#291-361) uses literals with too many digits:
        - res += xi * 0x000000000000000004985f67696bf748000 (contracts/Power.sol#328)
Power.fixedExp(uint256,uint8) (contracts/Power.sol#291-361) uses literals with too many digits:
        - res += xi * 0x0000000000000000003dea12ea99e498000 (contracts/Power.sol#330)
Power.fixedExp(uint256,uint8) (contracts/Power.sol#291-361) uses literals with too many digits:
        - res += xi * 0x000000000000000000031880f2214b6e000 (contracts/Power.sol#332)
Power.fixedExp(uint256,uint8) (contracts/Power.sol#291-361) uses literals with too many digits:
        - res += xi * 0x0000000000000000000025bcff56eb36000 (contracts/Power.sol#334)
Power.fixedExp(uint256,uint8) (contracts/Power.sol#291-361) uses literals with too many digits:
        - res += xi * 0x0000000000000000000001b722e10ab1000 (contracts/Power.sol#336)
Power.fixedExp(uint256,uint8) (contracts/Power.sol#291-361) uses literals with too many digits:
        - res += xi * 0x00000000000000000000001317c70077000 (contracts/Power.sol#338)
Power.fixedExp(uint256,uint8) (contracts/Power.sol#291-361) uses literals with too many digits:
        - res += xi * 0x000000000000000000000000cba84aafa00 (contracts/Power.sol#340)
Power.fixedExp(uint256,uint8) (contracts/Power.sol#291-361) uses literals with too many digits:
        - res += xi * 0x0000000000000000000000000082573a0a00 (contracts/Power.sol#342)
Power.fixedExp(uint256,uint8) (contracts/Power.sol#291-361) uses literals with too many digits:
        - res += xi * 0x0000000000000000000000000005035ad900 (contracts/Power.sol#344)
Power.fixedExp(uint256,uint8) (contracts/Power.sol#291-361) uses literals with too many digits:
        - res += xi * 0x00000000000000000000000000002f881b00 (contracts/Power.sol#346)
Power.fixedExp(uint256,uint8) (contracts/Power.sol#291-361) uses literals with too many digits:
        - res += xi * 0x000000000000000000000000000001b29340 (contracts/Power.sol#348)
Power.fixedExp(uint256,uint8) (contracts/Power.sol#291-361) uses literals with too many digits:
        - res += xi * 0x0000000000000000000000000000000efc40 (contracts/Power.sol#350)
Power.fixedExp(uint256,uint8) (contracts/Power.sol#291-361) uses literals with too many digits:
        - res += xi * 0x00000000000000000000000000000000007fe0 (contracts/Power.sol#352)
Power.fixedExp(uint256,uint8) (contracts/Power.sol#291-361) uses literals with too many digits:
        - res += xi * 0x00000000000000000000000000000000000420 (contracts/Power.sol#354)
Power.fixedExp(uint256,uint8) (contracts/Power.sol#291-361) uses literals with too many digits:
        - res += xi * 0x00000000000000000000000000000000000021 (contracts/Power.sol#356)
Power.fixedExp(uint256,uint8) (contracts/Power.sol#291-361) uses literals with too many digits:
        - res += xi * 0x00000000000000000000000000000000000001 (contracts/Power.sol#358)
Power.fixedExp(uint256,uint8) (contracts/Power.sol#291-361) uses literals with too many digits:
        - res / 0x688589cc0e9505e2f2fee5580000000 + _x + (ONE << _precision) (contracts/Power.sol#360)
BancorBondingCurve.slitherConstructorConstantVariables() (contracts/BancorBondingCurve.sol#16-161) uses literals with too many digits:
        - MAX_WEIGHT = 1000000 (contracts/Power.sol#19)
BancorBondingCurve.slitherConstructorConstantVariables() (contracts/BancorBondingCurve.sol#16-161) uses literals with too many digits:
        - FIXED_1 = 0x080000000000000000000000000000000 (contracts/Power.sol#27)
BancorBondingCurve.slitherConstructorConstantVariables() (contracts/BancorBondingCurve.sol#16-161) uses literals with too many digits:
        - FIXED_2 = 0x100000000000000000000000000000000 (contracts/Power.sol#28)
BancorBondingCurve.slitherConstructorConstantVariables() (contracts/BancorBondingCurve.sol#16-161) uses literals with too many digits:
        - MAX_RESERVE_RATIO = 1000000 (contracts/BancorBondingCurve.sol#18)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits

Power.MAX_WEIGHT (contracts/Power.sol#19) is never used in BancorBondingCurve (contracts/BancorBondingCurve.sol#16-161)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variable

calculatePriceForNTokens(uint32,uint256,uint32,uint32) should be declared external:
        - BancorBondingCurve.calculatePriceForNTokens(uint32,uint256,uint32,uint32) (contracts/BancorBondingCurve.sol#38-64)
calculatePurchaseReturn(uint32,uint256,uint32,uint256) should be declared external:
        - BancorBondingCurve.calculatePurchaseReturn(uint32,uint256,uint32,uint256) (contracts/BancorBondingCurve.sol#81-109)
calculateSaleReturn(uint32,uint256,uint32,uint32) should be declared external:
        - BancorBondingCurve.calculateSaleReturn(uint32,uint256,uint32,uint32) (contracts/BancorBondingCurve.sol#126-159)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

Power.sol

```
Power.ln(uint256,uint256) (contracts/Power.sol#205-230) performs a multiplication on the result of a division:
        -x = _numerator * FIXED_1 / _denominator (contracts/Power.sol#209)
        -z = (x * x) / FIXED_1 (contracts/Power.sol#221)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply

Different versions of Solidity is used:
        - Version used: ['^0.8.0', '^0.8.2']
        - ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/proxy/utils/Initializable.sol#4)
        - ^0.8.2 (contracts/Power.sol#3)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used

Power.findPositionInMaxExpArray(uint256) (contracts/Power.sol#263-282) is never used and should be removed
Power.fixedExp(uint256,uint8) (contracts/Power.sol#291-361) is never used and should be removed
Power.floorLog2(uint256) (contracts/Power.sol#235-256) is never used and should be removed
Power.ln(uint256,uint256) (contracts/Power.sol#205-230) is never used and should be removed
Power.power(uint256,uint256,uint32,uint32) (contracts/Power.sol#192-196) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/proxy/utils/Initializable.sol#4) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.2 (contracts/Power.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
solc-0.8.9 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Function Power.__Power_init() (contracts/Power.sol#43-174) is not in mixedCase
Parameter Power.power(uint256,uint256,uint32,uint32)._baseN (contracts/Power.sol#192) is not in mixedCase
Parameter Power.power(uint256,uint256,uint32,uint32)._baseD (contracts/Power.sol#192) is not in mixedCase
Parameter Power.power(uint256,uint256,uint32,uint32)._expN (contracts/Power.sol#192) is not in mixedCase
Parameter Power.power(uint256,uint256,uint32,uint32)._expD (contracts/Power.sol#192) is not in mixedCase
Parameter Power.ln(uint256,uint256)._numerator (contracts/Power.sol#205) is not in mixedCase
Parameter Power.ln(uint256,uint256)._denominator (contracts/Power.sol#205) is not in mixedCase
Parameter Power.floorLog2(uint256)._n (contracts/Power.sol#235) is not in mixedCase
Parameter Power.findPositionInMaxExpArray(uint256)._x (contracts/Power.sol#263) is not in mixedCase
Parameter Power.fixedExp(uint256,uint8)._x (contracts/Power.sol#291) is not in mixedCase
Parameter Power.fixedExp(uint256,uint8)._precision (contracts/Power.sol#291) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Power.fixedExp(uint256,uint8) (contracts/Power.sol#291-361) uses literals with too many digits:
        - res += xi * 0x03442c4e6074a82f1797f72ac0000000 (contracts/Power.sol#296)
Power.fixedExp(uint256,uint8) (contracts/Power.sol#291-361) uses literals with too many digits:
        - res += xi * 0x0116b96f757c380fb287fd0e40000000 (contracts/Power.sol#298)
Power.fixedExp(uint256,uint8) (contracts/Power.sol#291-361) uses literals with too many digits:
        - res += xi * 0x0045ae5bdd5f0e03eca1ff4390000000 (contracts/Power.sol#300)
Power.fixedExp(uint256,uint8) (contracts/Power.sol#291-361) uses literals with too many digits:
        - res += xi * 0x000defabf91302cd95b9ffda50000000 (contracts/Power.sol#302)
Power.fixedExp(uint256,uint8) (contracts/Power.sol#291-361) uses literals with too many digits:
        - res += xi * 0x0002529ca9832b22439efff9b8000000 (contracts/Power.sol#304)
Power.fixedExp(uint256,uint8) (contracts/Power.sol#291-361) uses literals with too many digits:
        - res += xi * 0x000054f1cf12bd04e516b6da88000000 (contracts/Power.sol#306)
Power.fixedExp(uint256,uint8) (contracts/Power.sol#291-361) uses literals with too many digits:
        - res += xi * 0x00000a9e39e257a09ca2d6db51000000 (contracts/Power.sol#308)
Power.fixedExp(uint256,uint8) (contracts/Power.sol#291-361) uses literals with too many digits:
        - res += xi * 0x0000012e066e7b839fa050c309000000 (contracts/Power.sol#310)
Power.fixedExp(uint256,uint8) (contracts/Power.sol#291-361) uses literals with too many digits:
        - res += xi * 0x0000001e33d7d926c329a1ad1a800000 (contracts/Power.sol#312)
Power.fixedExp(uint256,uint8) (contracts/Power.sol#291-361) uses literals with too many digits:
        - res += xi * 0x00000002bee513bdb4a6b19b5f800000 (contracts/Power.sol#314)
Power.fixedExp(uint256,uint8) (contracts/Power.sol#291-361) uses literals with too many digits:
        - res += xi * 0x000000003a9316fa79b88eccf2a00000 (contracts/Power.sol#316)
Power.fixedExp(uint256,uint8) (contracts/Power.sol#291-361) uses literals with too many digits:
        - res += xi * 0x00000000048177ebe1fa812375200000 (contracts/Power.sol#318)
Power.fixedExp(uint256,uint8) (contracts/Power.sol#291-361) uses literals with too many digits:
        - res += xi * 0x00000000005263fe90242dcbacf00000 (contracts/Power.sol#320)
Power.fixedExp(uint256,uint8) (contracts/Power.sol#291-361) uses literals with too many digits:
        - res += xi * 0x000000000057e22099c030d94100000 (contracts/Power.sol#322)
Power.fixedExp(uint256,uint8) (contracts/Power.sol#291-361) uses literals with too many digits:
        - res += xi * 0x0000000000057e22099c030d9410000 (contracts/Power.sol#324)
Power.fixedExp(uint256,uint8) (contracts/Power.sol#291-361) uses literals with too many digits:
        - res += xi * 0x00000000000052b6b54569976310000 (contracts/Power.sol#326)
Power.fixedExp(uint256,uint8) (contracts/Power.sol#291-361) uses literals with too many digits:
        - res += xi * 0x00000000000004985f67696bf748000 (contracts/Power.sol#328)
Power.fixedExp(uint256,uint8) (contracts/Power.sol#291-361) uses literals with too many digits:
        - res += xi * 0x000000000000003dea12ea99e498000 (contracts/Power.sol#330)
Power.fixedExp(uint256,uint8) (contracts/Power.sol#291-361) uses literals with too many digits:
        - res += xi * 0x00000000000000031880f2214b6e000 (contracts/Power.sol#332)
Power.fixedExp(uint256,uint8) (contracts/Power.sol#291-361) uses literals with too many digits:
        - res += xi * 0x000000000000000025bcff56eb36000 (contracts/Power.sol#334)
Power.fixedExp(uint256,uint8) (contracts/Power.sol#291-361) uses literals with too many digits:
        - res += xi * 0x000000000000000001b722e10ab1000 (contracts/Power.sol#336)
Power.fixedExp(uint256,uint8) (contracts/Power.sol#291-361) uses literals with too many digits:
        - res += xi * 0x0000000000000000001317c70077000 (contracts/Power.sol#338)
Power.fixedExp(uint256,uint8) (contracts/Power.sol#291-361) uses literals with too many digits:
        - res += xi * 0x00000000000000000000cba84aafa00 (contracts/Power.sol#340)
Power.fixedExp(uint256,uint8) (contracts/Power.sol#291-361) uses literals with too many digits:
        - res += xi * 0x00000000000000000000082573a0a00 (contracts/Power.sol#342)
Power.fixedExp(uint256,uint8) (contracts/Power.sol#291-361) uses literals with too many digits:
        - res += xi * 0x00000000000000000000005035ad900 (contracts/Power.sol#344)
Power.fixedExp(uint256,uint8) (contracts/Power.sol#291-361) uses literals with too many digits:
        - res += xi * 0x000000000000000000000002f881b00 (contracts/Power.sol#346)
Power.fixedExp(uint256,uint8) (contracts/Power.sol#291-361) uses literals with too many digits:
        - res += xi * 0x0000000000000000000000001b29340 (contracts/Power.sol#348)
Power.fixedExp(uint256,uint8) (contracts/Power.sol#291-361) uses literals with too many digits:
        - res += xi * 0x00000000000000000000000000efc40 (contracts/Power.sol#350)
Power.fixedExp(uint256,uint8) (contracts/Power.sol#291-361) uses literals with too many digits:
        - res += xi * 0x0000000000000000000000000007fe0 (contracts/Power.sol#352)
Power.fixedExp(uint256,uint8) (contracts/Power.sol#291-361) uses literals with too many digits:
        - res += xi * 0x0000000000000000000000000000420 (contracts/Power.sol#354)
Power.fixedExp(uint256,uint8) (contracts/Power.sol#291-361) uses literals with too many digits:
        - res += xi * 0x0000000000000000000000000000021 (contracts/Power.sol#356)
Power.fixedExp(uint256,uint8) (contracts/Power.sol#291-361) uses literals with too many digits:
        - res += xi * 0x0000000000000000000000000000001 (contracts/Power.sol#358)
Power.fixedExp(uint256,uint8) (contracts/Power.sol#291-361) uses literals with too many digits:
        - res / 0x688589cc0e9505e2f2fee5580000000 + _x + (ONE << _precision) (contracts/Power.sol#360)
Power.slitherConstructorConstantVariables() (contracts/Power.sol#15-362) uses literals with too many digits:
        - MAX_WEIGHT = 1000000 (contracts/Power.sol#19)
Power.slitherConstructorConstantVariables() (contracts/Power.sol#15-362) uses literals with too many digits:
        - FIXED_1 = 0x080000000000000000000000000000000 (contracts/Power.sol#27)
Power.slitherConstructorConstantVariables() (contracts/Power.sol#15-362) uses literals with too many digits:
        - FIXED_2 = 0x100000000000000000000000000000000 (contracts/Power.sol#28)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits

Power.MAX_WEIGHT (contracts/Power.sol#19) is never used in Power (contracts/Power.sol#15-362)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variable

__Power_init() should be declared external:
        - Power.__Power_init() (contracts/Power.sol#43-174)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

ProductNft.sol

```
ERC721._checkOnERC721Received(address,address,uint256,bytes) (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#369-390) ignores return value by IERC721Receiver(to).onERC721Received(_msgSender(),from,tokenId,_data) (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#376-386)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return

Variable 'ERC721._checkOnERC721Received(address,address,uint256,bytes).retval (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#376)' in ERC721._checkOnERC721Received(address,address,uint256,bytes) (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#369-390) potentially used before declaration: retval == IERC721Receiver.onERC721Received.selector (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#377)
Variable 'ERC721._checkOnERC721Received(address,address,uint256,bytes).reason (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#378)' in ERC721._checkOnERC721Received(address,address,uint256,bytes) (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#369-390) potentially used before declaration: reason.length == 0 (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#379)
Variable 'ERC721._checkOnERC721Received(address,address,uint256,bytes).reason (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#378)' in ERC721._checkOnERC721Received(address,address,uint256,bytes) (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#369-390) potentially used before declaration: revert(uint256,uint256)(32 + reason,mload(uint256)(reason)) (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#383)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#pre-declaration-usage-of-local-variables

ERC721._checkOnERC721Received(address,address,uint256,bytes) (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#369-390) uses assembly
        - INLINE ASM (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#382-384)
Address.isContract(address) (node_modules/@openzeppelin/contracts/utils/Address.sol#26-36) uses assembly
        - INLINE ASM (node_modules/@openzeppelin/contracts/utils/Address.sol#32-34)
Address.verifyCallResult(bool,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#195-215) uses assembly
        - INLINE ASM (node_modules/@openzeppelin/contracts/utils/Address.sol#207-210)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

ProductNft.mint(address,uint256) (contracts/ProductNft.sol#126-129) compares to a boolean constant:
        -require(bool,string)(minters[_msgSender()] == true,permission denied) (contracts/ProductNft.sol#125)
ProductNft.mintBatch(address,uint256,uint256) (contracts/ProductNft.sol#140-149) compares to a boolean constant:
        -require(bool,string)(minters[_msgSender()] == true,permission denied) (contracts/ProductNft.sol#141)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#boolean-equality

Different versions of Solidity is used:
        - Version used: ['^0.8.0', '^0.8.9']
        - ^0.8.0 (node_modules/@openzeppelin/contracts/access/Ownable.sol#3)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#3)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC721/IERC721.sol#3)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC721/IERC721Receiver.sol#3)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC721/extensions/IERC721Enumerable.sol#3)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC721/extensions/IERC721Enumerable.sol#3)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC721/extensions/IERC721Metadata.sol#3)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/utils/Address.sol#3)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/utils/Context.sol#3)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/utils/Strings.sol#3)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/utils/introspection/ERC165.sol#3)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/utils/introspection/IERC165.sol#3)
        - ^0.8.9 (contracts/ProductNft.sol#2)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used

Address.functionCall(address,bytes) (node_modules/@openzeppelin/contracts/utils/Address.sol#79-81) is never used and should be removed
Address.functionCall(address,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#89-95) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (node_modules/@openzeppelin/contracts/utils/Address.sol#105-114) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#122-133) is never used and should be removed
Address.functionDelegateCall(address,bytes) (node_modules/@openzeppelin/contracts/utils/Address.sol#168-170) is never used and should be removed
Address.functionDelegateCall(address,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#178-187) is never used and should be removed
Address.functionStaticCall(address,bytes) (node_modules/@openzeppelin/contracts/utils/Address.sol#141-143) is never used and should be removed
Address.functionStaticCall(address,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#151-160) is never used and should be removed
Address.sendValue(address,uint256) (node_modules/@openzeppelin/contracts/utils/Address.sol#56-59) is never used and should be removed
Address.verifyCallResult(bool,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#195-215) is never used and should be removed
Context._msgData() (node_modules/@openzeppelin/contracts/utils/Context.sol#20-22) is never used and should be removed
ERC721._baseURI() (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#104-106) is never used and should be removed
ERC721._burn(uint256) (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#304-316) is never used and should be removed
Strings.toHexString(uint256) (node_modules/@openzeppelin/contracts/utils/Strings.sol#39-50) is never used and should be removed
Strings.toHexString(uint256,uint256) (node_modules/@openzeppelin/contracts/utils/Strings.sol#55-65) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/access/Ownable.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC721/IERC721.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6

ERC721._checkOnERC721Received(address,address,uint256,bytes) (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#369-390) ignores return value by IERC721Receiver(to).onERC721Received(_msgSender(),from,tokenId,_data) (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#376-386)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return

Variable 'ERC721._checkOnERC721Received(address,address,uint256,bytes).retval (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#376)' in ERC721._checkOnERC721Received(address,address,uint256,bytes) (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#369-390) potentially used before declaration: retval == IERC721Receiver.onERC721Received.selector (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#377)
Variable 'ERC721._checkOnERC721Received(address,address,uint256,bytes).reason (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#378)' in ERC721._checkOnERC721Received(address,address,uint256,bytes) (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#369-390) potentially used before declaration: reason.length == 0 (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#379)
Variable 'ERC721._checkOnERC721Received(address,address,uint256,bytes).reason (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#378)' in ERC721._checkOnERC721Received(address,address,uint256,bytes) (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#369-390) potentially used before declaration: revert(uint256,uint256)(32 + reason,mload(uint256)(reason)) (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#383)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#pre-declaration-usage-of-local-variables

ERC721._checkOnERC721Received(address,address,uint256,bytes) (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#369-390) uses assembly
        - INLINE ASM (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#382-384)
Address.isContract(address) (node_modules/@openzeppelin/contracts/utils/Address.sol#26-36) uses assembly
        - INLINE ASM (node_modules/@openzeppelin/contracts/utils/Address.sol#32-34)
Address.verifyCallResult(bool,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#195-215) uses assembly
        - INLINE ASM (node_modules/@openzeppelin/contracts/utils/Address.sol#207-210)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

ProductNft.mint(address,uint256) (contracts/ProductNft.sol#126-129) compares to a boolean constant:
        -require(bool,string)(minters[_msgSender()] == true,permission denied) (contracts/ProductNft.sol#125)
ProductNft.mintBatch(address,uint256,uint256) (contracts/ProductNft.sol#140-149) compares to a boolean constant:
        -require(bool,string)(minters[_msgSender()] == true,permission denied) (contracts/ProductNft.sol#141)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#boolean-equality

Different versions of Solidity is used:
        - Version used: ['^0.8.0', '^0.8.9']
        - ^0.8.0 (node_modules/@openzeppelin/contracts/access/Ownable.sol#3)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#3)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC721/IERC721.sol#3)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC721/IERC721Receiver.sol#3)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC721/extensions/IERC721Enumerable.sol#3)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC721/extensions/IERC721Enumerable.sol#3)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC721/extensions/IERC721Metadata.sol#3)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/utils/Address.sol#3)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/utils/Context.sol#3)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/utils/Strings.sol#3)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/utils/introspection/ERC165.sol#3)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/utils/introspection/IERC165.sol#3)
        - ^0.8.9 (contracts/ProductNft.sol#2)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used

Address.functionCall(address,bytes) (node_modules/@openzeppelin/contracts/utils/Address.sol#79-81) is never used and should be removed
Address.functionCall(address,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#89-95) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (node_modules/@openzeppelin/contracts/utils/Address.sol#105-114) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#122-133) is never used and should be removed
Address.functionDelegateCall(address,bytes) (node_modules/@openzeppelin/contracts/utils/Address.sol#168-170) is never used and should be removed
Address.functionDelegateCall(address,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#178-187) is never used and should be removed
Address.functionStaticCall(address,bytes) (node_modules/@openzeppelin/contracts/utils/Address.sol#141-143) is never used and should be removed
Address.functionStaticCall(address,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#151-160) is never used and should be removed
Address.sendValue(address,uint256) (node_modules/@openzeppelin/contracts/utils/Address.sol#56-59) is never used and should be removed
Address.verifyCallResult(bool,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#195-215) is never used and should be removed
Context._msgData() (node_modules/@openzeppelin/contracts/utils/Context.sol#20-22) is never used and should be removed
ERC721._baseURI() (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#104-106) is never used and should be removed
ERC721._burn(uint256) (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#304-316) is never used and should be removed
Strings.toHexString(uint256) (node_modules/@openzeppelin/contracts/utils/Strings.sol#39-50) is never used and should be removed
Strings.toHexString(uint256,uint256) (node_modules/@openzeppelin/contracts/utils/Strings.sol#55-65) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/access/Ownable.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC721/IERC721.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
```

ProductTokenCore.sol

```
OwnableUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol#82) shadows:
        - ContextUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#31)
PausableUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/security/PausableUpgradeable.sol#97) shadows:
        - ContextUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#31)
ERC20Upgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#462) shadows:
        - ContextUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#31)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variable-shadowing

Power.ln(uint256,uint256) (contracts/Power.sol#206-230) performs a multiplication on the result of a division:
        -x = _numerator * FIXED_1 / _denominator (contracts/Power.sol#209)
        -z = (x * x) / FIXED_1 (contracts/Power.sol#221)
ProductTokenCore._buyReturn(uint256) (contracts/ProductTokenCore.sol#807-821) performs a multiplication on the result of a division:
        -value = _amountReserve * FEE_DIVIDER / (FEE_DIVIDER + FEE_RATE_IN_BUY) (contracts/ProductTokenCore.sol#810)
        -fee = value * FEE_RATE_IN_BUY / FEE_DIVIDER (contracts/ProductTokenCore.sol#812)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply

Reentrancy in ProductTokenCore._tradein(uint32) (contracts/ProductTokenCore.sol#458-472):
        External calls:
        - nft.mintBatch(_msgSender(),tradeinCount,amount_) (contracts/ProductTokenCore.sol#466)
        State variables written after the call(s):
        - tradeinCount = tradeinCount + amount_ (contracts/ProductTokenCore.sol#468)
        - tradeinReserveBalance = tradeinReserveBalance + tradinReturn (contracts/ProductTokenCore.sol#469)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1

ERC721._checkOnERC721Received(address,address,uint256,bytes) (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#369-390) ignores return value by IERC721Receiver(to).onERC721Received(_msgSender(),from,tokenId,_data) (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#376-386)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return

ProductTokenCore.initialize(string,string,address,address,uint32,uint32,uint32,uint256)._name (contracts/ProductTokenCore.sol#175) shadows:
        - ERC20Upgradeable._name (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#53) (state variable)
ProductTokenCore.initialize(string,string,address,address,uint32,uint32,uint32,uint256)._symbol (contracts/ProductTokenCore.sol#176) shadows:
        - ERC20Upgradeable._symbol (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#54) (state variable)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
```

AUTOMATED TESTING

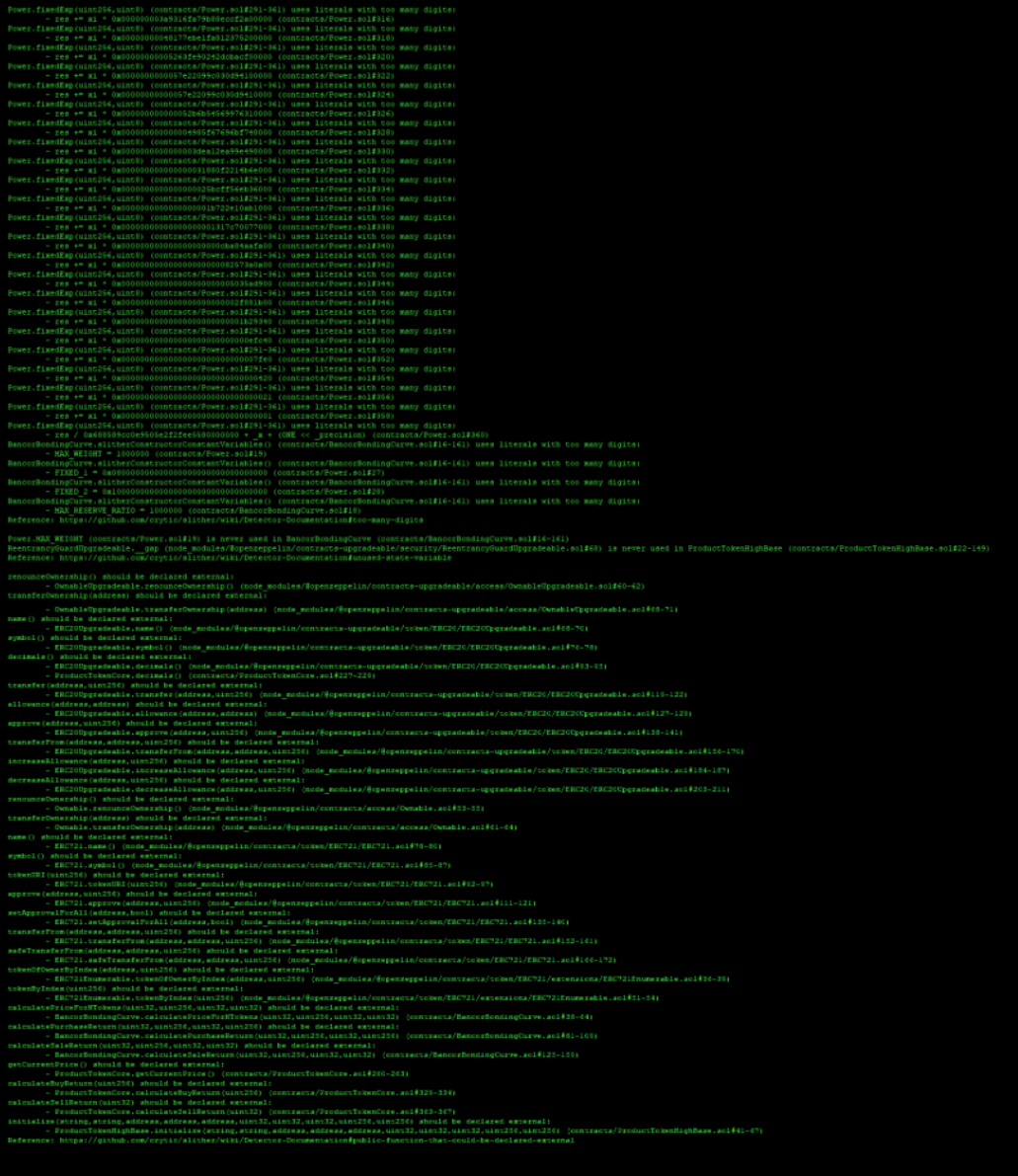
```
Variable 'ERC721._checkOnERC721Received(address,address,uint256,bytes).retval (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#376)' in ERC721._checkOnERC721Received(address,address,uint256,bytes) (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#369-390) potentially used before declaration: retval == IERC721Receiver.onERC721Received.selector (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#377)
Variable 'ERC721._checkOnERC721Received(address,address,uint256,bytes).reason (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#378)' in ERC721._checkOnERC721Received(address,address,uint256,bytes) (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#369-390) potentially used before declaration: reason.length == 0 (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#379)
Variable 'ERC721._checkOnERC721Received(address,address,uint256,bytes).reason (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#378)' in ERC721._checkOnERC721Received(address,address,uint256,bytes) (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#369-390) potentially used before declaration: revert(uint256,uint256)(32 + reason,mload(uint256)(reason)) (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#383)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#pre-declaration-usage-of-local-variables

Reentrancy in ProductTokenCore._tradein(uint32) (contracts/ProductTokenCore.sol#453-472):
        External calls:
        - nft.mintBatch(_msgSender(),tradeinCount,amount_) (contracts/ProductTokenCore.sol#466)
        Event emitted after the call(s):
        - Tradein(_msgSender(),amount_,tradinReturn,fee) (contracts/ProductTokenCore.sol#471)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3

ProductTokenCore.updateEndTime(uint32,uint256) (contracts/ProductTokenCore.sol#618-632) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(endTime_ > now256(),invalid endTime) (contracts/ProductTokenCore.sol#620)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp

ERC721._checkOnERC721Received(address,address,uint256,bytes) (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#369-390) uses assembly
        - INLINE ASM (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#382-384)
Address.isContract(address) (node_modules/@openzeppelin/contracts/utils/Address.sol#26-36) uses assembly
        - INLINE ASM (node_modules/@openzeppelin/contracts/utils/Address.sol#32-34)
Address.verifyCallResult(bool,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#195-215) uses assembly
        - INLINE ASM (node_modules/@openzeppelin/contracts/utils/Address.sol#207-210)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

ProductNft.mint(address,uint256) (contracts/ProductNft.sol#124-129) compares to a boolean constant:
        -require(bool,string)(minters[_msgSender()] == true,permission denied) (contracts/ProductNft.sol#125)
ProductNft.mintBatch(address,uint256,uint256) (contracts/ProductNft.sol#140-149) compares to a boolean constant:
        -require(bool,string)(minters[_msgSender()] == true,permission denied) (contracts/ProductNft.sol#141)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#boolean-equality

Different versions of Solidity is used:
        - Version used: ['^0.8.0', '^0.8.2', '^0.8.9']
        - ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/proxy/utils/Initializable.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/security/PausableUpgradeable.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/security/ReentrancyGuardUpgradeable.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/IERC20Upgradeable.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/extensions/IERC20MetadataUpgradeable.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/utils/math/SafeMathUpgradeable.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/access/Ownable.sol#3)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/IERC20.sol#3)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#3)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC721/IERC721.sol#3)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC721/IERC721Receiver.sol#3)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC721/extensions/ERC721Enumerable.sol#3)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC721/extensions/IERC721Enumerable.sol#3)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC721/extensions/IERC721Metadata.sol#3)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/utils/Address.sol#3)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/utils/Context.sol#3)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/utils/Strings.sol#3)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/utils/introspection/ERC165.sol#3)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/utils/introspection/IERC165.sol#3)
        - ^0.8.2 (contracts/BancorBondingCurve.sol#3)
        - ^0.8.2 (contracts/Power.sol#3)
        - ^0.8.9 (contracts/ProductNft.sol#2)
        - ^0.8.9 (contracts/ProductTokenCore.sol#3)

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used

Address.functionCall(address,bytes) (node_modules/@openzeppelin/contracts/utils/Address.sol#79-81) is never used and should be removed
Address.functionCall(address,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#89-95) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (node_modules/@openzeppelin/contracts/utils/Address.sol#108-114) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#122-133) is never used and should be removed
Address.functionDelegateCall(address,bytes) (node_modules/@openzeppelin/contracts/utils/Address.sol#168-170) is never used and should be removed
Address.functionDelegateCall(address,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#178-187) is never used and should be removed
Address.functionStaticCall(address,bytes) (node_modules/@openzeppelin/contracts/utils/Address.sol#141-143) is never used and should be removed
Address.functionStaticCall(address,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#151-160) is never used and should be removed
Address.sendValue(address,uint256) (node_modules/@openzeppelin/contracts/utils/Address.sol#54-59) is never used and should be removed
Address.verifyCallResult(bool,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#195-215) is never used and should be removed
Context._msgData() (node_modules/@openzeppelin/contracts/utils/Context.sol#20-22) is never used and should be removed
ContextUpgradeable.__Context_init() (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#18-20) is never used and should be removed
ContextUpgradeable._msgData() (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#28-30) is never used and should be removed
ERC20Upgradeable._burn(address,uint256) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#281-296) is never used and should be removed
ERC20Upgradeable._mint(address,uint256) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#258-268) is never used and should be removed
ERC721._baseURI() (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#104-106) is never used and should be removed
ERC721._burn(uint256) (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#304-316) is never used and should be removed
ProductTokenCore._buy(uint256) (contracts/ProductTokenCore.sol#399-417) is never used and should be removed
ProductTokenCore._sell(uint32) (contracts/ProductTokenCore.sol#428-445) is never used and should be removed
ProductTokenCore._tradein(uint32) (contracts/ProductTokenCore.sol#453-472) is never used and should be removed
ProductTokenCore._tradinReturn(uint32) (contracts/ProductTokenCore.sol#377-386) is never used and should be removed
ProductTokenCore._updatePlateformFee(uint256) (contracts/ProductTokenCore.sol#488-490) is never used and should be removed
ProductTokenCore._updateSupplierFee(uint256) (contracts/ProductTokenCore.sol#479-481) is never used and should be removed
SafeMathUpgradeable.div(uint256,uint256,string) (node_modules/@openzeppelin/contracts-upgradeable/utils/math/SafeMathUpgradeable.sol#191-200) is never used and should be removed
SafeMathUpgradeable.mod(uint256,uint256) (node_modules/@openzeppelin/contracts-upgradeable/utils/math/SafeMathUpgradeable.sol#151-153) is never used and should be removed
SafeMathUpgradeable.mod(uint256,uint256,string) (node_modules/@openzeppelin/contracts-upgradeable/utils/math/SafeMathUpgradeable.sol#217-226) is never used and should be removed
SafeMathUpgradeable.sub(uint256,uint256,string) (node_modules/@openzeppelin/contracts-upgradeable/utils/math/SafeMathUpgradeable.sol#168-177) is never used and should be removed
SafeMathUpgradeable.tryAdd(uint256,uint256) (node_modules/@openzeppelin/contracts-upgradeable/utils/math/SafeMathUpgradeable.sol#22-28) is never used and should be removed
SafeMathUpgradeable.tryDiv(uint256,uint256) (node_modules/@openzeppelin/contracts-upgradeable/utils/math/SafeMathUpgradeable.sol#64-69) is never used and should be removed
SafeMathUpgradeable.tryMod(uint256,uint256) (node_modules/@openzeppelin/contracts-upgradeable/utils/math/SafeMathUpgradeable.sol#76-81) is never used and should be removed
SafeMathUpgradeable.tryMul(uint256,uint256) (node_modules/@openzeppelin/contracts-upgradeable/utils/math/SafeMathUpgradeable.sol#47-57) is never used and should be removed
SafeMathUpgradeable.trySub(uint256,uint256) (node_modules/@openzeppelin/contracts-upgradeable/utils/math/SafeMathUpgradeable.sol#35-40) is never used and should be removed
Strings.toHexString(uint256) (node_modules/@openzeppelin/contracts/utils/Strings.sol#39-50) is never used and should be removed
Strings.toHexString(uint256,uint256) (node_modules/@openzeppelin/contracts/utils/Strings.sol#55-65) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol#4) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/proxy/utils/Initializable.sol#4) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/security/PausableUpgradeable.sol#4) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/security/ReentrancyGuardUpgradeable.sol#4) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#4) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/IERC20Upgradeable.sol#4) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/extensions/IERC20MetadataUpgradeable.sol#4) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#4) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/utils/math/SafeMathUpgradeable.sol#4) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/access/Ownable.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/IERC20.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC721/IERC721.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC721/IERC721Receiver.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC721/extensions/ERC721Enumerable.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC721/extensions/IERC721Enumerable.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC721/extensions/IERC721Metadata.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/utils/Address.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/utils/Context.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/utils/Strings.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/utils/introspection/ERC165.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/utils/introspection/IERC165.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.2 (contracts/BancorBondingCurve.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.2 (contracts/Power.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.9 (contracts/ProductNft.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6

Pragma version^0.8.9 (contracts/ProductTokenCore.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
solc-0.8.9 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) (node_modules/@openzeppelin/contracts/utils/Address.sol#54-59):
        - (success) = recipient.call{value: amount}() (node_modules/@openzeppelin/contracts/utils/Address.sol#57)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#122-133):
        - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#131)
Low level call in Address.functionStaticCall(address,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#151-160):
        - (success,returndata) = target.staticcall(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#158)
Low level call in Address.functionDelegateCall(address,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#178-187):
        - (success,returndata) = target.delegatecall(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#185)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Function OwnableUpgradeable.__Ownable_init() (node_modules/@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol#29-32) is not in mixedCase
Function OwnableUpgradeable.__Ownable_init_unchained() (node_modules/@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol#34-36) is not in mixedCase
Variable OwnableUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol#82) is not in mixedCase
Function PausableUpgradeable.__Pausable_init() (node_modules/@openzeppelin/contracts-upgradeable/security/PausableUpgradeable.sol#34-37) is not in mixedCase
Function PausableUpgradeable.__Pausable_init_unchained() (node_modules/@openzeppelin/contracts-upgradeable/security/PausableUpgradeable.sol#39-41) is not in mixedCase
Variable PausableUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/security/PausableUpgradeable.sol#97) is not in mixedCase
Function ReentrancyGuardUpgradeable.__ReentrancyGuard_init() (node_modules/@openzeppelin/contracts-upgradeable/security/ReentrancyGuardUpgradeable.sol#40-42) is not in mixedCase
Function ReentrancyGuardUpgradeable.__ReentrancyGuard_init_unchained() (node_modules/@openzeppelin/contracts-upgradeable/security/ReentrancyGuardUpgradeable.sol#44-46) is not in mixedCase
Variable ReentrancyGuardUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/security/ReentrancyGuardUpgradeable.sol#68) is not in mixedCase
Function ERC20Upgradeable.__ERC20_init(string,string) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#55-58) is not in mixedCase
Function ERC20Upgradeable.__ERC20_init_unchained(string,string) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#60-63) is not in mixedCase
Variable ERC20Upgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#362) is not in mixedCase
Function ContextUpgradeable.__Context_init() (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#18-20) is not in mixedCase
Function ContextUpgradeable.__Context_init_unchained() (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#22-23) is not in mixedCase
Variable ContextUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#31) is not in mixedCase
Parameter ERC721.safeTransferFrom(address,address,uint256,bytes)._data (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#181) is not in mixedCase
Parameter BancorBondingCurve.calculatePriceForNTokens(uint32,uint256,uint32,uint32)._supply (contracts/BancorBondingCurve.sol#39) is not in mixedCase
Parameter BancorBondingCurve.calculatePriceForNTokens(uint32,uint256,uint32,uint32)._reserveBalance (contracts/BancorBondingCurve.sol#40) is not in mixedCase
Parameter BancorBondingCurve.calculatePriceForNTokens(uint32,uint256,uint32,uint32)._reserveRatio (contracts/BancorBondingCurve.sol#41) is not in mixedCase
Parameter BancorBondingCurve.calculatePriceForNTokens(uint32,uint256,uint32,uint32)._amount (contracts/BancorBondingCurve.sol#42) is not in mixedCase
Parameter BancorBondingCurve.calculatePurchaseReturn(uint32,uint256,uint32,uint256)._supply (contracts/BancorBondingCurve.sol#82) is not in mixedCase
Parameter BancorBondingCurve.calculatePurchaseReturn(uint32,uint256,uint32,uint256)._reserveBalance (contracts/BancorBondingCurve.sol#83) is not in mixedCase
Parameter BancorBondingCurve.calculatePurchaseReturn(uint32,uint256,uint32,uint256)._reserveRatio (contracts/BancorBondingCurve.sol#84) is not in mixedCase
Parameter BancorBondingCurve.calculatePurchaseReturn(uint32,uint256,uint32,uint256)._depositAmount (contracts/BancorBondingCurve.sol#85) is not in mixedCase
Parameter BancorBondingCurve.calculateSaleReturn(uint32,uint256,uint32,uint32)._supply (contracts/BancorBondingCurve.sol#126) is not in mixedCase
Parameter BancorBondingCurve.calculateSaleReturn(uint32,uint256,uint32,uint32)._reserveBalance (contracts/BancorBondingCurve.sol#127) is not in mixedCase
Parameter BancorBondingCurve.calculateSaleReturn(uint32,uint256,uint32,uint32)._reserveRatio (contracts/BancorBondingCurve.sol#128) is not in mixedCase
Parameter BancorBondingCurve.calculateSaleReturn(uint32,uint256,uint32,uint32)._sellAmount (contracts/BancorBondingCurve.sol#129) is not in mixedCase
Function Power._Power_init() (contracts/Power.sol#43-174) is not in mixedCase
```

```
Parameter Power.power(uint256,uint256,uint32,uint32)._baseN (contracts/Power.sol#192) is not in mixedCase
Parameter Power.power(uint256,uint256,uint32,uint32)._baseD (contracts/Power.sol#192) is not in mixedCase
Parameter Power.power(uint256,uint256,uint32,uint32)._expN (contracts/Power.sol#192) is not in mixedCase
Parameter Power.power(uint256,uint256,uint32,uint32)._expD (contracts/Power.sol#192) is not in mixedCase
Parameter Power.ln(uint256,uint256)._numerator (contracts/Power.sol#205) is not in mixedCase
Parameter Power.ln(uint256,uint256)._denominator (contracts/Power.sol#205) is not in mixedCase
Parameter Power.floorLog2(uint256)._n (contracts/Power.sol#235) is not in mixedCase
Parameter Power.findPositionInMaxExpArray(uint256)._x (contracts/Power.sol#263) is not in mixedCase
Parameter Power.fixedExp(uint256,uint8)._x (contracts/Power.sol#291) is not in mixedCase
Parameter Power.fixedExp(uint256,uint8)._precision (contracts/Power.sol#291) is not in mixedCase
Parameter ProductTokenCore.initialize(string,string,address,address,uint32,uint32,uint32,uint256)._name (contracts/ProductTokenCore.sol#175) is not in mixedCase
Parameter ProductTokenCore.initialize(string,string,address,address,uint32,uint32,uint32,uint256)._symbol (contracts/ProductTokenCore.sol#176) is not in mixedCase
Parameter ProductTokenCore.initialize(string,string,address,address,uint32,uint32,uint32,uint256)._bondingCurve (contracts/ProductTokenCore.sol#177) is not in mixedCase
Parameter ProductTokenCore.initialize(string,string,address,address,uint32,uint32,uint32,uint256)._productNft (contracts/ProductTokenCore.sol#178) is not in mixedCase
Parameter ProductTokenCore.initialize(string,string,address,address,uint32,uint32,uint32,uint256)._reserveRatio (contracts/ProductTokenCore.sol#179) is not in mixedCase
Parameter ProductTokenCore.initialize(string,string,address,address,uint32,uint32,uint32,uint256)._maxTokenCount (contracts/ProductTokenCore.sol#180) is not in mixedCase
Parameter ProductTokenCore.initialize(string,string,address,address,uint32,uint32,uint32,uint256)._supplyOffset (contracts/ProductTokenCore.sol#181) is not in mixedCase
Parameter ProductTokenCore.initialize(string,string,address,address,uint32,uint32,uint32,uint256)._baseReserve (contracts/ProductTokenCore.sol#182) is not in mixedCase
Function ProductTokenCore.__ProductToken_init_unchained(address,address,uint32,uint32,uint32,uint256) (contracts/ProductTokenCore.sol#203-220) is not in mixedCase
Parameter ProductTokenCore.__ProductToken_init_unchained(address,address,uint32,uint32,uint32,uint256)._bondingCurve (contracts/ProductTokenCore.sol#204) is not in mixedCase
Parameter ProductTokenCore.__ProductToken_init_unchained(address,address,uint32,uint32,uint32,uint256)._productNft (contracts/ProductTokenCore.sol#205) is not in mixedCase
Parameter ProductTokenCore.__ProductToken_init_unchained(address,address,uint32,uint32,uint32,uint256)._reserveRatio (contracts/ProductTokenCore.sol#206) is not in mixedCase
Parameter ProductTokenCore.__ProductToken_init_unchained(address,address,uint32,uint32,uint32,uint256)._maxTokenCount (contracts/ProductTokenCore.sol#207) is not in mixedCase
Parameter ProductTokenCore.__ProductToken_init_unchained(address,address,uint32,uint32,uint32,uint256)._supplyOffset (contracts/ProductTokenCore.sol#208) is not in mixedCase
Parameter ProductTokenCore.__ProductToken_init_unchained(address,address,uint32,uint32,uint32,uint256)._baseReserve (contracts/ProductTokenCore.sol#209) is not in mixedCase
Parameter ProductTokenCore.getPriceForN(uint32)._amountProduct (contracts/ProductTokenCore.sol#272) is not in mixedCase
Parameter ProductTokenCore.calculateBuyReturn(uint256)._amountReserve (contracts/ProductTokenCore.sol#329) is not in mixedCase
Parameter ProductTokenCore.calculateSellReturn(uint32)._amountProduct (contracts/ProductTokenCore.sol#363) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Power.fixedExp(uint256,uint8) (contracts/Power.sol#291-361) uses literals with too many digits:
        - res += xi * 0x03442c4e6074a82f1797f72ac0000000 (contracts/Power.sol#296)
Power.fixedExp(uint256,uint8) (contracts/Power.sol#291-361) uses literals with too many digits:
        - res += xi * 0x0116b96f757c380fb287fd0e40000000 (contracts/Power.sol#298)
Power.fixedExp(uint256,uint8) (contracts/Power.sol#291-361) uses literals with too many digits:
        - res += xi * 0x0045ae5bdd5f0e03eca1ff4390000000 (contracts/Power.sol#300)
Power.fixedExp(uint256,uint8) (contracts/Power.sol#291-361) uses literals with too many digits:
        - res += xi * 0x000defabf91302cd95b9ffda50000000 (contracts/Power.sol#302)
Power.fixedExp(uint256,uint8) (contracts/Power.sol#291-361) uses literals with too many digits:
        - res += xi * 0x0002529ca9832b22439efff9b8000000 (contracts/Power.sol#304)
Power.fixedExp(uint256,uint8) (contracts/Power.sol#291-361) uses literals with too many digits:
        - res += xi * 0x000054f1cf12bd04e516b6da88000000 (contracts/Power.sol#306)
Power.fixedExp(uint256,uint8) (contracts/Power.sol#291-361) uses literals with too many digits:
        - res += xi * 0x00000a9e39e257a09ca2d6db51000000 (contracts/Power.sol#308)
Power.fixedExp(uint256,uint8) (contracts/Power.sol#291-361) uses literals with too many digits:
        - res += xi * 0x0000012e066e7b839fa050c309000000 (contracts/Power.sol#310)
Power.fixedExp(uint256,uint8) (contracts/Power.sol#291-361) uses literals with too many digits:
        - res += xi * 0x0000001e33d7d926c329a1ad1a800000 (contracts/Power.sol#312)
Power.fixedExp(uint256,uint8) (contracts/Power.sol#291-361) uses literals with too many digits:
        - res += xi * 0x00000002bee513bdb4a6b19b5f800000 (contracts/Power.sol#314)
Power.fixedExp(uint256,uint8) (contracts/Power.sol#291-361) uses literals with too many digits:
        - res += xi * 0x000000003a9316fa79b88eccf2a00000 (contracts/Power.sol#316)
Power.fixedExp(uint256,uint8) (contracts/Power.sol#291-361) uses literals with too many digits:
        - res += xi * 0x00000000048177ebe1fa812375200000 (contracts/Power.sol#318)
Power.fixedExp(uint256,uint8) (contracts/Power.sol#291-361) uses literals with too many digits:
        - res += xi * 0x00000000005263fe90242dcbacf00000 (contracts/Power.sol#320)
Power.fixedExp(uint256,uint8) (contracts/Power.sol#291-361) uses literals with too many digits:
        - res += xi * 0x0000000000057e22099c030d94100000 (contracts/Power.sol#322)
Power.fixedExp(uint256,uint8) (contracts/Power.sol#291-361) uses literals with too many digits:
        - res += xi * 0x00000000000057e22099c030d9410000 (contracts/Power.sol#324)
Power.fixedExp(uint256,uint8) (contracts/Power.sol#291-361) uses literals with too many digits:
        - res += xi * 0x0000000000000052b6b54569976310000 (contracts/Power.sol#326)
Power.fixedExp(uint256,uint8) (contracts/Power.sol#291-361) uses literals with too many digits:

        - res += xi * 0x00000000000000004985f67696bf748000 (contracts/Power.sol#328)
Power.fixedExp(uint256,uint8) (contracts/Power.sol#291-361) uses literals with too many digits:
        - res += xi * 0x0000000000000003dea12ea99e498000 (contracts/Power.sol#330)
Power.fixedExp(uint256,uint8) (contracts/Power.sol#291-361) uses literals with too many digits:
        - res += xi * 0x000000000000000031880f2214b6e000 (contracts/Power.sol#332)
Power.fixedExp(uint256,uint8) (contracts/Power.sol#291-361) uses literals with too many digits:
        - res += xi * 0x0000000000000000025bcff56eb36000 (contracts/Power.sol#334)
Power.fixedExp(uint256,uint8) (contracts/Power.sol#291-361) uses literals with too many digits:
        - res += xi * 0x00000000000000000001b722e10ab1000 (contracts/Power.sol#336)
Power.fixedExp(uint256,uint8) (contracts/Power.sol#291-361) uses literals with too many digits:
        - res += xi * 0x000000000000000000001317c70077000 (contracts/Power.sol#338)
Power.fixedExp(uint256,uint8) (contracts/Power.sol#291-361) uses literals with too many digits:
        - res += xi * 0x0000000000000000000000cba84aafa00 (contracts/Power.sol#340)
Power.fixedExp(uint256,uint8) (contracts/Power.sol#291-361) uses literals with too many digits:
        - res += xi * 0x000000000000000000000082573a0a00 (contracts/Power.sol#342)
Power.fixedExp(uint256,uint8) (contracts/Power.sol#291-361) uses literals with too many digits:
        - res += xi * 0x000000000000000000000005035ad900 (contracts/Power.sol#344)
Power.fixedExp(uint256,uint8) (contracts/Power.sol#291-361) uses literals with too many digits:
        - res += xi * 0x0000000000000000000000002f881b00 (contracts/Power.sol#346)
Power.fixedExp(uint256,uint8) (contracts/Power.sol#291-361) uses literals with too many digits:
        - res += xi * 0x00000000000000000000000001b29340 (contracts/Power.sol#348)
Power.fixedExp(uint256,uint8) (contracts/Power.sol#291-361) uses literals with too many digits:
        - res += xi * 0x000000000000000000000000000efc40 (contracts/Power.sol#350)
Power.fixedExp(uint256,uint8) (contracts/Power.sol#291-361) uses literals with too many digits:
        - res += xi * 0x00000000000000000000000000007fe0 (contracts/Power.sol#352)
Power.fixedExp(uint256,uint8) (contracts/Power.sol#291-361) uses literals with too many digits:
        - res += xi * 0x00000000000000000000000000000420 (contracts/Power.sol#354)
Power.fixedExp(uint256,uint8) (contracts/Power.sol#291-361) uses literals with too many digits:
        - res += xi * 0x00000000000000000000000000000021 (contracts/Power.sol#356)
Power.fixedExp(uint256,uint8) (contracts/Power.sol#291-361) uses literals with too many digits:
        - res += xi * 0x00000000000000000000000000000001 (contracts/Power.sol#358)
Power.fixedExp(uint256,uint8) (contracts/Power.sol#291-361) uses literals with too many digits:
        - res / 0x688589cc0e9505e2f2fee5580000000 + _x + (ONE << _precision) (contracts/Power.sol#360)
BancorBondingCurve.slitherConstructorConstantVariables() (contracts/BancorBondingCurve.sol#16-161) uses literals with too many digits:
        - MAX_WEIGHT = 1000000 (contracts/Power.sol#19)
BancorBondingCurve.slitherConstructorConstantVariables() (contracts/BancorBondingCurve.sol#16-161) uses literals with too many digits:
        - FIXED_1 = 0x080000000000000000000000000000000 (contracts/Power.sol#27)
BancorBondingCurve.slitherConstructorConstantVariables() (contracts/BancorBondingCurve.sol#16-161) uses literals with too many digits:
        - FIXED_2 = 0x100000000000000000000000000000000 (contracts/Power.sol#28)
BancorBondingCurve.slitherConstructorConstantVariables() (contracts/BancorBondingCurve.sol#16-161) uses literals with too many digits:
        - MAX_RESERVE_RATIO = 1000000 (contracts/BancorBondingCurve.sol#18)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits

Power.MAX_WEIGHT (contracts/Power.sol#19) is never used in BancorBondingCurve (contracts/BancorBondingCurve.sol#16-161)
ReentrancyGuardUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/security/ReentrancyGuardUpgradeable.sol#68) is never used in ProductTokenCore (contracts/ProductTokenCore.sol#23-635)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variable

renounceOwnership() should be declared external:
        - OwnableUpgradeable.renounceOwnership() (node_modules/@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol#60-62)
transferOwnership(address) should be declared external:
        - OwnableUpgradeable.transferOwnership(address) (node_modules/@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol#68-71)
name() should be declared external:
        - ERC20Upgradeable.name() (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#68-70)
symbol() should be declared external:
        - ERC20Upgradeable.symbol() (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#76-78)
decimals() should be declared external:
        - ERC20Upgradeable.decimals() (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#93-95)
        - ProductTokenCore.decimals() (contracts/ProductTokenCore.sol#227-229)
balanceOf(address) should be declared external:
        - ERC20Upgradeable.balanceOf(address) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#107-109)
transfer(address,uint256) should be declared external:
        - ERC20Upgradeable.transfer(address,uint256) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#119-122)
allowance(address,address) should be declared external:

        - ERC20Upgradeable.allowance(address,address) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#127-129)
approve(address,uint256) should be declared external:
        - ERC20Upgradeable.approve(address,uint256) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#138-141)
transferFrom(address,address,uint256) should be declared external:
        - ERC20Upgradeable.transferFrom(address,address,uint256) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#156-170)
increaseAllowance(address,uint256) should be declared external:
        - ERC20Upgradeable.increaseAllowance(address,uint256) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#184-187)
decreaseAllowance(address,uint256) should be declared external:
        - ERC20Upgradeable.decreaseAllowance(address,uint256) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#203-211)
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (node_modules/@openzeppelin/contracts/access/Ownable.sol#53-55)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (node_modules/@openzeppelin/contracts/access/Ownable.sol#61-64)
name() should be declared external:
        - ERC721.name() (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#78-80)
symbol() should be declared external:
        - ERC721.symbol() (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#85-87)
tokenURI(uint256) should be declared external:
        - ERC721.tokenURI(uint256) (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#92-97)
approve(address,uint256) should be declared external:
        - ERC721.approve(address,uint256) (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#111-121)
setApprovalForAll(address,bool) should be declared external:
        - ERC721.setApprovalForAll(address,bool) (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#135-140)
transferFrom(address,address,uint256) should be declared external:
        - ERC721.transferFrom(address,address,uint256) (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#152-161)
safeTransferFrom(address,address,uint256) should be declared external:
        - ERC721.safeTransferFrom(address,address,uint256) (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#166-172)
tokenOfOwnerByIndex(address,uint256) should be declared external:
        - ERC721Enumerable.tokenOfOwnerByIndex(address,uint256) (node_modules/@openzeppelin/contracts/token/ERC721/extensions/ERC721Enumerable.sol#36-39)
tokenByIndex(uint256) should be declared external:
        - ERC721Enumerable.tokenByIndex(uint256) (node_modules/@openzeppelin/contracts/token/ERC721/extensions/ERC721Enumerable.sol#51-54)
calculatePriceForNTokens(uint32,uint256,uint32,uint32) should be declared external:
        - BancorBondingCurve.calculatePriceForNTokens(uint32,uint256,uint32,uint32) (contracts/BancorBondingCurve.sol#38-64)
calculatePurchaseReturn(uint32,uint256,uint32,uint256) should be declared external:
        - BancorBondingCurve.calculatePurchaseReturn(uint32,uint256,uint32,uint256) (contracts/BancorBondingCurve.sol#81-109)
calculateSaleReturn(uint32,uint256,uint32,uint32) should be declared external:
        - BancorBondingCurve.calculateSaleReturn(uint32,uint256,uint32,uint32) (contracts/BancorBondingCurve.sol#125-159)
initialize(string,string,address,address,uint32,uint32,uint32,uint256) should be declared external:
        - ProductTokenCore.initialize(string,string,address,address,uint32,uint32,uint32,uint256) (contracts/ProductTokenCore.sol#174-190)
getCurrentPrice() should be declared external:
        - ProductTokenCore.getCurrentPrice() (contracts/ProductTokenCore.sol#260-263)
calculateBuyReturn(uint256) should be declared external:
        - ProductTokenCore.calculateBuyReturn(uint256) (contracts/ProductTokenCore.sol#329-334)
calculateSellReturn(uint32) should be declared external:
        - ProductTokenCore.calculateSellReturn(uint32) (contracts/ProductTokenCore.sol#363-367)
updateEndTime(uint32,uint256) should be declared external:
        - ProductTokenCore.updateEndTime(uint32,uint256) (contracts/ProductTokenCore.sol#618-632)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

ProductTokenHighBase.sol

```
OwnableUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol#92) shadows:
        - ContextUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#31)
PausableUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/security/PausableUpgradeable.sol#97) shadows:
        - ContextUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#31)
ERC20Upgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#362) shadows:
        - ContextUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#31)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variable-shadowing

Power.ln(uint256,uint256) (contracts/Power.sol#205-230) performs a multiplication on the result of a division:
        -x = _numerator * FIXED_1 / _denominator (contracts/Power.sol#209)
        -z = z_ * x / FIXED_1 (contracts/Power.sol#221)
ProductTokenCore._buyReturn(uint256) (contracts/ProductTokenCore.sol#307-321) performs a multiplication on the result of a division:
        -value = _amountReserve * FEE_DIVIDER / (FEE_DIVIDER + FEE_RATE_IN_BUY) (contracts/ProductTokenCore.sol#310)
        -fee = value * FEE_RATE_IN_BUY / FEE_DIVIDER (contracts/ProductTokenCore.sol#312)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply

Reentrancy in ProductTokenCore._tradein(uint32) (contracts/ProductTokenCore.sol#453-472):
        External calls:
        - nft.mintBatch(_msgSender(),tradeinCount,amount_) (contracts/ProductTokenCore.sol#466)
        State variables written after the call(s):
        - tradeinCount = tradeinCount + amount_ (contracts/ProductTokenCore.sol#468)
        - tradeinReserveBalance = tradeinReserveBalance + tradinReturn (contracts/ProductTokenCore.sol#469)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1

ERC721._checkOnERC721Received(address,address,uint256,bytes) (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#369-390) ignores return value by IERC721Receiver(to).onERC721Received(_msgSender(),from,tokenId,_data) (node_modu
le/@openzeppelin/contracts/token/ERC721/ERC721.sol#376-386)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return

ProductTokenCore.initialize(string,string,address,address,uint32,uint32,uint32,uint256)._name (contracts/ProductTokenCore.sol#179) shadows:
        - ERC20Upgradeable._name (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#49) (state variable)
ProductTokenCore.initialize(string,string,address,address,uint32,uint32,uint32,uint256)._symbol (contracts/ProductTokenCore.sol#176) shadows:
        - ERC20Upgradeable._symbol (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#44) (state variable)
ProductTokenHighBase.initialize(string,string,address,address,address,uint32,uint32,uint32,uint256)._name (contracts/ProductTokenHighBase.sol#42) shadows:
        - ERC20Upgradeable._name (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#49) (state variable)
ProductTokenHighBase.initialize(string,string,address,address,address,uint32,uint32,uint32,uint256)._symbol (contracts/ProductTokenHighBase.sol#43) shadows:
        - ERC20Upgradeable._symbol (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#44) (state variable)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing

ProductTokenHighBase.initialize(string,string,address,address,address,uint32,uint32,uint32,uint256)._high (contracts/ProductTokenHighBase.sol#44) lacks a zero-check on :
        - HIGH = _high (contracts/ProductTokenHighBase.sol#53)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

Variable 'ERC721._checkOnERC721Received(address,address,uint256,bytes).retval (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#376)' in ERC721._checkOnERC721Received(address,address,uint256,bytes) (node_modules/@openzeppelin
/contracts/token/ERC721/ERC721.sol#369-390) potentially used before declaration: retval == IERC721Receiver.onERC721Received.selector (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#377)
Variable 'ERC721._checkOnERC721Received(address,address,uint256,bytes).reason (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#378)' in ERC721._checkOnERC721Received(address,address,uint256,bytes) (node_modules/@openzeppelin
/contracts/token/ERC721/ERC721.sol#369-390) potentially used before declaration: reason.length == 0 (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#379)
Variable 'ERC721._checkOnERC721Received(address,address,uint256,bytes).reason (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#378)' in ERC721._checkOnERC721Received(address,address,uint256,bytes) (node_modules/@openzeppelin
/contracts/token/ERC721/ERC721.sol#369-390) potentially used before declaration: revert(uint256,uint256)(32 + reason,mload(uint256)(reason)) (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#383)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#pre-declaration-usage-of-local-variables

Reentrancy in ProductTokenHighBase.buy(uint256) (contracts/ProductTokenHighBase.sol#78-90):
        External calls:
        - transferFromHighToken(_msgSender(),address(this),maxPrice_) (contracts/ProductTokenHighBase.sol#84)
                - SafeERC20Upgradeable.safeTransferFrom(IERC20Upgradeable(HIGH),from_,to_,value_) (contracts/ProductTokenHighBase.sol#147)
                - returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/utils/SafeERC20Upgradeable.sol#93)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#132)
        External calls sending eth:
        - transferFromHighToken(_msgSender(),address(this),maxPrice_) (contracts/ProductTokenHighBase.sol#84)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#132)
        State variables written after the call(s):
        - change = _buy(maxPrice_) (contracts/ProductTokenHighBase.sol#84)
                - _balances[account] += amount (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#264)
        - change = _buy(maxPrice_) (contracts/ProductTokenHighBase.sol#84)

                - _totalSupply += amount (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#263)
        - change = _buy(maxPrice_) (contracts/ProductTokenHighBase.sol#84)
                - plateformFee = plateformFee + fee_ (contracts/ProductTokenCore.sol#489)
        - change = _buy(maxPrice_) (contracts/ProductTokenHighBase.sol#84)
                - reserveBalance = reserveBalance + price (contracts/ProductTokenCore.sol#413)
        - change = _buy(maxPrice_) (contracts/ProductTokenHighBase.sol#84)
                - supplier.amount = supplier.amount + fee_ (contracts/ProductTokenCore.sol#490)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2

Reentrancy in ProductTokenCore._tradein(uint32) (contracts/ProductTokenCore.sol#453-472):
        External calls:
        - nft.mintBatch(_msgSender(),tradeinCount,amount_) (contracts/ProductTokenCore.sol#466)
        Event emitted after the call(s):
        - Tradein(_msgSender(),amount_,tradinReturn,fee) (contracts/ProductTokenCore.sol#471)
Reentrancy in ProductTokenHighBase.buy(uint256) (contracts/ProductTokenHighBase.sol#78-90):
        External calls:
        - transferFromHighToken(_msgSender(),address(this),maxPrice_) (contracts/ProductTokenHighBase.sol#84)
                - SafeERC20Upgradeable.safeTransferFrom(IERC20Upgradeable(HIGH),from_,to_,value_) (contracts/ProductTokenHighBase.sol#147)
                - returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/utils/SafeERC20Upgradeable.sol#93)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#132)
        External calls sending eth:
        - transferFromHighToken(_msgSender(),address(this),maxPrice_) (contracts/ProductTokenHighBase.sol#84)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#132)
        Event emitted after the call(s):
        - Buy(_msgSender(),price,fee) (contracts/ProductTokenCore.sol#415)
                - change = _buy(maxPrice_) (contracts/ProductTokenHighBase.sol#86)
        - Transfer(address(0),account,amount) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#265)
                - change = _buy(maxPrice_) (contracts/ProductTokenHighBase.sol#86)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3

ProductTokenCore.updateEndTime(uint32,uint256) (contracts/ProductTokenCore.sol#615-632) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(endTime_ > now256(),invalid endTime) (contracts/ProductTokenCore.sol#620)
ProductTokenHighBase.buy(uint256) (contracts/ProductTokenHighBase.sol#78-90) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(now256() < endTime[FEATURE_ENDTIME_BUY],sale is expire) (contracts/ProductTokenHighBase.sol#80)
ProductTokenHighBase.sell(uint32) (contracts/ProductTokenHighBase.sol#100-109) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(now256() < endTime[FEATURE_ENDTIME_SELL],sale is expire) (contracts/ProductTokenHighBase.sol#102)
ProductTokenHighBase.tradein(uint32) (contracts/ProductTokenHighBase.sol#116-123) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(now256() < endTime[FEATURE_ENDTIME_TRADEIN],sale is expire) (contracts/ProductTokenHighBase.sol#118)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp

AddressUpgradeable.isContract(address) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#27-37) uses assembly
        - INLINE ASM (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#33-35)
AddressUpgradeable.verifyCallResult(bool,bytes,string) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#189-209) uses assembly
        - INLINE ASM (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#201-204)
ERC721._checkOnERC721Received(address,address,uint256,bytes) (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#369-390) uses assembly
        - INLINE ASM (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#382-384)
Address.isContract(address) (node_modules/@openzeppelin/contracts/utils/Address.sol#26-36) uses assembly
        - INLINE ASM (node_modules/@openzeppelin/contracts/utils/Address.sol#32-34)
Address.verifyCallResult(bool,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#195-215) uses assembly
        - INLINE ASM (node_modules/@openzeppelin/contracts/utils/Address.sol#207-210)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

ProductNft.mint(address,uint256) (contracts/ProductNft.sol#124-129) compares to a boolean constant:
        -require(bool,string)(minters[_msgSender()] == true,permission denied) (contracts/ProductNft.sol#125)
ProductNft.mintBatch(address,uint256,uint256) (contracts/ProductNft.sol#140-149) compares to a boolean constant:
        -require(bool,string)(minters[_msgSender()] == true,permission denied) (contracts/ProductNft.sol#141)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#boolean-equality

Different versions of Solidity is used:
        - Version used: ['^0.8.0', '^0.8.2', '^0.8.9']
        - ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/proxy/utils/Initializable.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/security/PausableUpgradeable.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/security/ReentrancyGuardUpgradeable.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/IERC20Upgradeable.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/extensions/IERC20MetadataUpgradeable.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/utils/SafeERC20Upgradeable.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/utils/math/SafeMathUpgradeable.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/access/Ownable.sol#3)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#3)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#3)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC721/IERC721.sol#3)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC721/IERC721Receiver.sol#3)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC721/extensions/ERC721Enumerable.sol#3)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC721/extensions/IERC721Enumerable.sol#3)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC721/extensions/IERC721Metadata.sol#3)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/utils/Address.sol#3)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/utils/Context.sol#3)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/utils/Strings.sol#3)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/utils/introspection/ERC165.sol#3)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/utils/introspection/IERC165.sol#3)
        - ^0.8.2 (contracts/BancorBondingCurve.sol#3)
        - ^0.8.2 (contracts/Power.sol#3)
        - ^0.8.9 (contracts/ProductNft.sol#2)
        - ^0.8.9 (contracts/ProductTokenCore.sol#3)
        - ^0.8.9 (contracts/ProductTokenHighBase.sol#3)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used
```

```
Address.functionCall(address,bytes) (node_modules/@openzeppelin/contracts/utils/Address.sol#79-81) is never used and should be removed
Address.functionCall(address,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#89-95) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (node_modules/@openzeppelin/contracts/utils/Address.sol#108-114) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#122-133) is never used and should be removed
Address.functionDelegateCall(address,bytes) (node_modules/@openzeppelin/contracts/utils/Address.sol#168-170) is never used and should be removed
Address.functionDelegateCall(address,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#178-187) is never used and should be removed
Address.functionStaticCall(address,bytes) (node_modules/@openzeppelin/contracts/utils/Address.sol#141-143) is never used and should be removed
Address.functionStaticCall(address,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#151-160) is never used and should be removed
Address.sendValue(address,uint256) (node_modules/@openzeppelin/contracts/utils/Address.sol#54-59) is never used and should be removed
Address.verifyCallResult(bool,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#195-215) is never used and should be removed
AddressUpgradeable.functionCall(address,bytes) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#80-82) is never used and should be removed
AddressUpgradeable.functionCallWithValue(address,bytes,uint256) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#109-115) is never used and should be removed
AddressUpgradeable.functionStaticCall(address,bytes) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#142-144) is never used and should be removed
AddressUpgradeable.functionStaticCall(address,bytes,string) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#152-161) is never used and should be removed
AddressUpgradeable.sendValue(address,uint256) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#55-60) is never used and should be removed
Context._msgData() (node_modules/@openzeppelin/contracts/utils/Context.sol#20-22) is never used and should be removed
ContextUpgradeable.__Context_init() (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#18-20) is never used and should be removed
ContextUpgradeable._msgData() (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#28-30) is never used and should be removed
ERC721._baseURI() (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#104-106) is never used and should be removed
ERC721._burn(uint256) (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#304-316) is never used and should be removed
ProductTokenCore._claim(uint256) (contracts/ProductTokenCore.sol#577) is never used and should be removed
SafeERC20Upgradeable.safeApprove(IERC20Upgradeable,address,uint256) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/utils/SafeERC20Upgradeable.sol#45-58) is never used and should be removed
SafeERC20Upgradeable.safeDecreaseAllowance(IERC20Upgradeable,address,uint256) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/utils/SafeERC20Upgradeable.sol#69-80) is never used and should be removed
SafeERC20Upgradeable.safeIncreaseAllowance(IERC20Upgradeable,address,uint256) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/utils/SafeERC20Upgradeable.sol#60-67) is never used and should be removed
SafeMathUpgradeable.div(uint256,uint256,string) (node_modules/@openzeppelin/contracts-upgradeable/utils/math/SafeMathUpgradeable.sol#191-200) is never used and should be removed
SafeMathUpgradeable.mod(uint256,uint256) (node_modules/@openzeppelin/contracts-upgradeable/utils/math/SafeMathUpgradeable.sol#151-153) is never used and should be removed
SafeMathUpgradeable.mod(uint256,uint256,string) (node_modules/@openzeppelin/contracts-upgradeable/utils/math/SafeMathUpgradeable.sol#217-226) is never used and should be removed
SafeMathUpgradeable.sub(uint256,uint256,string) (node_modules/@openzeppelin/contracts-upgradeable/utils/math/SafeMathUpgradeable.sol#168-177) is never used and should be removed
SafeMathUpgradeable.tryAdd(uint256,uint256) (node_modules/@openzeppelin/contracts-upgradeable/utils/math/SafeMathUpgradeable.sol#22-28) is never used and should be removed
SafeMathUpgradeable.tryDiv(uint256,uint256) (node_modules/@openzeppelin/contracts-upgradeable/utils/math/SafeMathUpgradeable.sol#64-69) is never used and should be removed
SafeMathUpgradeable.tryMod(uint256,uint256) (node_modules/@openzeppelin/contracts-upgradeable/utils/math/SafeMathUpgradeable.sol#76-81) is never used and should be removed
SafeMathUpgradeable.tryMul(uint256,uint256) (node_modules/@openzeppelin/contracts-upgradeable/utils/math/SafeMathUpgradeable.sol#47-57) is never used and should be removed
SafeMathUpgradeable.trySub(uint256,uint256) (node_modules/@openzeppelin/contracts-upgradeable/utils/math/SafeMathUpgradeable.sol#35-40) is never used and should be removed
Strings.toHexString(uint256) (node_modules/@openzeppelin/contracts/utils/Strings.sol#39-50) is never used and should be removed
Strings.toHexString(uint256,uint256) (node_modules/@openzeppelin/contracts/utils/Strings.sol#55-65) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol#4) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/proxy/utils/Initializable.sol#4) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/security/PausableUpgradeable.sol#4) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/security/ReentrancyGuardUpgradeable.sol#4) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#4) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/IERC20Upgradeable.sol#4) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/extensions/IERC20MetadataUpgradeable.sol#4) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/utils/SafeERC20Upgradeable.sol#4) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#4) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#4) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/utils/math/SafeMathUpgradeable.sol#4) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/access/Ownable.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/IERC20.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC721/IERC721.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC721/IERC721Receiver.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC721/extensions/ERC721Enumerable.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC721/extensions/IERC721Enumerable.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC721/extensions/IERC721Metadata.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/utils/Address.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/utils/Context.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/utils/Strings.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/utils/introspection/ERC165.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/utils/introspection/IERC165.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.2 (contracts/BancorBondingCurve.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6

Pragma version^0.8.2 (contracts/Power.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.9 (contracts/ProductNft.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.9 (contracts/ProductTokenCore.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.9 (contracts/ProductTokenHighBase.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
solc-0.8.9 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in AddressUpgradeable.sendValue(address,uint256) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#55-60):
        - (success) = recipient.call{value: amount}() (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#58)
Low level call in AddressUpgradeable.functionCallWithValue(address,bytes,uint256,string) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#123-134):
        - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#132)
Low level call in AddressUpgradeable.functionStaticCall(address,bytes,string) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#152-161):
        - (success,returndata) = target.staticcall(data) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#159)
Low level call in Address.sendValue(address,uint256) (node_modules/@openzeppelin/contracts/utils/Address.sol#54-59):
        - (success) = recipient.call{value: amount}() (node_modules/@openzeppelin/contracts/utils/Address.sol#57)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#122-133):
        - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#131)
Low level call in Address.functionStaticCall(address,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#151-160):
        - (success,returndata) = target.staticcall(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#158)
Low level call in Address.functionDelegateCall(address,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#178-187):
        - (success,returndata) = target.delegatecall(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#185)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Function OwnableUpgradeable.__Ownable_init() (node_modules/@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol#29-32) is not in mixedCase
Function OwnableUpgradeable.__Ownable_init_unchained() (node_modules/@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol#34-36) is not in mixedCase
Variable OwnableUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol#82) is not in mixedCase
Function PausableUpgradeable.__Pausable_init() (node_modules/@openzeppelin/contracts-upgradeable/security/PausableUpgradeable.sol#34-37) is not in mixedCase
Function PausableUpgradeable.__Pausable_init_unchained() (node_modules/@openzeppelin/contracts-upgradeable/security/PausableUpgradeable.sol#39-41) is not in mixedCase
Variable PausableUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/security/PausableUpgradeable.sol#97) is not in mixedCase
Function ReentrancyGuardUpgradeable.__ReentrancyGuard_init() (node_modules/@openzeppelin/contracts-upgradeable/security/ReentrancyGuardUpgradeable.sol#40-42) is not in mixedCase
Function ReentrancyGuardUpgradeable.__ReentrancyGuard_init_unchained() (node_modules/@openzeppelin/contracts-upgradeable/security/ReentrancyGuardUpgradeable.sol#44-46) is not in mixedCase
Variable ReentrancyGuardUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/security/ReentrancyGuardUpgradeable.sol#68) is not in mixedCase
Function ERC20Upgradeable.__ERC20_init(string,string) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#55-58) is not in mixedCase
Function ERC20Upgradeable.__ERC20_init_unchained(string,string) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#60-63) is not in mixedCase
Variable ERC20Upgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#362) is not in mixedCase
Function ContextUpgradeable.__Context_init() (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#18-20) is not in mixedCase
Function ContextUpgradeable.__Context_init_unchained() (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#22-23) is not in mixedCase
Variable ContextUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#31) is not in mixedCase
Parameter ERC721.safeTransferFrom(address,address,uint256,bytes)._data (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#181) is not in mixedCase
Parameter BancorBondingCurve.calculatePriceForNTokens(uint32,uint256,uint32,uint32)._supply (contracts/BancorBondingCurve.sol#39) is not in mixedCase
Parameter BancorBondingCurve.calculatePriceForNTokens(uint32,uint256,uint32,uint32)._reserveBalance (contracts/BancorBondingCurve.sol#40) is not in mixedCase
Parameter BancorBondingCurve.calculatePriceForNTokens(uint32,uint256,uint32,uint32)._reserveRatio (contracts/BancorBondingCurve.sol#41) is not in mixedCase
Parameter BancorBondingCurve.calculatePriceForNTokens(uint32,uint256,uint32,uint32)._amount (contracts/BancorBondingCurve.sol#42) is not in mixedCase
Parameter BancorBondingCurve.calculatePurchaseReturn(uint32,uint256,uint32,uint256)._supply (contracts/BancorBondingCurve.sol#82) is not in mixedCase
Parameter BancorBondingCurve.calculatePurchaseReturn(uint32,uint256,uint32,uint256)._reserveBalance (contracts/BancorBondingCurve.sol#83) is not in mixedCase
Parameter BancorBondingCurve.calculatePurchaseReturn(uint32,uint256,uint32,uint256)._reserveRatio (contracts/BancorBondingCurve.sol#84) is not in mixedCase
Parameter BancorBondingCurve.calculatePurchaseReturn(uint32,uint256,uint32,uint256)._depositAmount (contracts/BancorBondingCurve.sol#85) is not in mixedCase
Parameter BancorBondingCurve.calculateSaleReturn(uint32,uint256,uint32,uint32)._supply (contracts/BancorBondingCurve.sol#126) is not in mixedCase
Parameter BancorBondingCurve.calculateSaleReturn(uint32,uint256,uint32,uint32)._reserveBalance (contracts/BancorBondingCurve.sol#127) is not in mixedCase
Parameter BancorBondingCurve.calculateSaleReturn(uint32,uint256,uint32,uint32)._reserveRatio (contracts/BancorBondingCurve.sol#128) is not in mixedCase
Parameter BancorBondingCurve.calculateSaleReturn(uint32,uint256,uint32,uint32)._sellAmount (contracts/BancorBondingCurve.sol#129) is not in mixedCase
Function Power.__Power_init() (contracts/Power.sol#43-174) is not in mixedCase
Parameter Power.power(uint256,uint256,uint32,uint32)._baseN (contracts/Power.sol#192) is not in mixedCase
Parameter Power.power(uint256,uint256,uint32,uint32)._baseD (contracts/Power.sol#192) is not in mixedCase
Parameter Power.power(uint256,uint256,uint32,uint32)._expN (contracts/Power.sol#192) is not in mixedCase
Parameter Power.power(uint256,uint256,uint32,uint32)._expD (contracts/Power.sol#192) is not in mixedCase
Parameter Power.ln(uint256,uint256)._numerator (contracts/Power.sol#205) is not in mixedCase
Parameter Power.ln(uint256,uint256)._denominator (contracts/Power.sol#205) is not in mixedCase
Parameter Power.floorLog2(uint256)._n (contracts/Power.sol#235) is not in mixedCase
Parameter Power.findPositionInMaxExpArray(uint256)._x (contracts/Power.sol#263) is not in mixedCase
Parameter Power.fixedExp(uint256,uint8)._x (contracts/Power.sol#291) is not in mixedCase
Parameter Power.fixedExp(uint256,uint8)._precision (contracts/Power.sol#291) is not in mixedCase
Parameter ProductTokenCore.initialize(string,string,address,address,uint32,uint32,uint32,uint256)._name (contracts/ProductTokenCore.sol#175) is not in mixedCase

Parameter ProductTokenCore.initialize(string,string,address,address,uint32,uint32,uint32,uint256)._symbol (contracts/ProductTokenCore.sol#176) is not in mixedCase
Parameter ProductTokenCore.initialize(string,string,address,address,uint32,uint32,uint32,uint256)._bondingCurve (contracts/ProductTokenCore.sol#177) is not in mixedCase
Parameter ProductTokenCore.initialize(string,string,address,address,uint32,uint32,uint32,uint256)._productNft (contracts/ProductTokenCore.sol#178) is not in mixedCase
Parameter ProductTokenCore.initialize(string,string,address,address,uint32,uint32,uint32,uint256)._reserveRatio (contracts/ProductTokenCore.sol#179) is not in mixedCase
Parameter ProductTokenCore.initialize(string,string,address,address,uint32,uint32,uint32,uint256)._maxTokenCount (contracts/ProductTokenCore.sol#180) is not in mixedCase
Parameter ProductTokenCore.initialize(string,string,address,address,uint32,uint32,uint32,uint256)._supplyOffset (contracts/ProductTokenCore.sol#181) is not in mixedCase
Parameter ProductTokenCore.initialize(string,string,address,address,uint32,uint32,uint32,uint256)._baseReserve (contracts/ProductTokenCore.sol#182) is not in mixedCase
Function ProductTokenCore.__ProductToken_init_unchained(address,address,uint32,uint32,uint32,uint256) (contracts/ProductTokenCore.sol#203-220) is not in mixedCase
Parameter ProductTokenCore.__ProductToken_init_unchained(address,address,uint32,uint32,uint32,uint256)._bondingCurve (contracts/ProductTokenCore.sol#204) is not in mixedCase
Parameter ProductTokenCore.__ProductToken_init_unchained(address,address,uint32,uint32,uint32,uint256)._productNft (contracts/ProductTokenCore.sol#205) is not in mixedCase
Parameter ProductTokenCore.__ProductToken_init_unchained(address,address,uint32,uint32,uint32,uint256)._reserveRatio (contracts/ProductTokenCore.sol#206) is not in mixedCase
Parameter ProductTokenCore.__ProductToken_init_unchained(address,address,uint32,uint32,uint32,uint256)._maxTokenCount (contracts/ProductTokenCore.sol#207) is not in mixedCase
Parameter ProductTokenCore.__ProductToken_init_unchained(address,address,uint32,uint32,uint32,uint256)._supplyOffset (contracts/ProductTokenCore.sol#208) is not in mixedCase
Parameter ProductTokenCore.__ProductToken_init_unchained(address,address,uint32,uint32,uint32,uint256)._baseReserve (contracts/ProductTokenCore.sol#209) is not in mixedCase
Parameter ProductTokenCore.getPriceForN(uint32)._amountProduct (contracts/ProductTokenCore.sol#272) is not in mixedCase
Parameter ProductTokenCore.calculateBuyReturn(uint256)._amountReserve (contracts/ProductTokenCore.sol#329) is not in mixedCase
Parameter ProductTokenCore.calculateSellReturn(uint32)._amountProduct (contracts/ProductTokenCore.sol#363) is not in mixedCase
Parameter ProductTokenHighBase.initialize(string,string,address,address,address,uint32,uint32,uint32,uint256,uint256)._name (contracts/ProductTokenHighBase.sol#42) is not in mixedCase
Parameter ProductTokenHighBase.initialize(string,string,address,address,address,uint32,uint32,uint32,uint256,uint256)._symbol (contracts/ProductTokenHighBase.sol#43) is not in mixedCase
Parameter ProductTokenHighBase.initialize(string,string,address,address,address,uint32,uint32,uint32,uint256,uint256)._high (contracts/ProductTokenHighBase.sol#44) is not in mixedCase
Parameter ProductTokenHighBase.initialize(string,string,address,address,address,uint32,uint32,uint32,uint256,uint256)._bondingCurve (contracts/ProductTokenHighBase.sol#45) is not in mixedCase
Parameter ProductTokenHighBase.initialize(string,string,address,address,address,uint32,uint32,uint32,uint256,uint256)._productNft (contracts/ProductTokenHighBase.sol#46) is not in mixedCase
Parameter ProductTokenHighBase.initialize(string,string,address,address,address,uint32,uint32,uint32,uint256,uint256)._reserveRatio (contracts/ProductTokenHighBase.sol#47) is not in mixedCase
Parameter ProductTokenHighBase.initialize(string,string,address,address,address,uint32,uint32,uint32,uint256,uint256)._maxTokenCount (contracts/ProductTokenHighBase.sol#48) is not in mixedCase
Parameter ProductTokenHighBase.initialize(string,string,address,address,address,uint32,uint32,uint32,uint256,uint256)._supplyOffset (contracts/ProductTokenHighBase.sol#49) is not in mixedCase
Parameter ProductTokenHighBase.initialize(string,string,address,address,address,uint32,uint32,uint32,uint256,uint256)._baseReserve (contracts/ProductTokenHighBase.sol#50) is not in mixedCase
Parameter ProductTokenHighBase.initialize(string,string,address,address,address,uint32,uint32,uint32,uint256,uint256)._endTime (contracts/ProductTokenHighBase.sol#51) is not in mixedCase
Variable ProductTokenHighBase.HIGH (contracts/ProductTokenHighBase.sol#25) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Power.fixedExp(uint256,uint8) (contracts/Power.sol#291-361) uses literals with too many digits:
        - res += xi * 0x03442c4e6074a82f1797f72ac0000000 (contracts/Power.sol#296)
Power.fixedExp(uint256,uint8) (contracts/Power.sol#291-361) uses literals with too many digits:
        - res += xi * 0x0116b96f757c380fb287fd0e40000000 (contracts/Power.sol#298)
Power.fixedExp(uint256,uint8) (contracts/Power.sol#291-361) uses literals with too many digits:
        - res += xi * 0x0045ae5bdd5f0e03eca1ff4390000000 (contracts/Power.sol#300)
Power.fixedExp(uint256,uint8) (contracts/Power.sol#291-361) uses literals with too many digits:
        - res += xi * 0x000defabf91302cd95b9ffda50000000 (contracts/Power.sol#302)
Power.fixedExp(uint256,uint8) (contracts/Power.sol#291-361) uses literals with too many digits:
        - res += xi * 0x0002529ca9832b22439efff9b8000000 (contracts/Power.sol#304)
Power.fixedExp(uint256,uint8) (contracts/Power.sol#291-361) uses literals with too many digits:
        - res += xi * 0x000054f1cf12bd04e516b6da88000000 (contracts/Power.sol#306)
Power.fixedExp(uint256,uint8) (contracts/Power.sol#291-361) uses literals with too many digits:
        - res += xi * 0x00000a9e39e257a09ca2d6db51000000 (contracts/Power.sol#308)
Power.fixedExp(uint256,uint8) (contracts/Power.sol#291-361) uses literals with too many digits:
        - res += xi * 0x0000012e066e7b839fa050c309000000 (contracts/Power.sol#310)
Power.fixedExp(uint256,uint8) (contracts/Power.sol#291-361) uses literals with too many digits:
        - res += xi * 0x0000001e33d7d926c329a1ad1a800000 (contracts/Power.sol#312)
Power.fixedExp(uint256,uint8) (contracts/Power.sol#291-361) uses literals with too many digits:
        - res += xi * 0x00000002bee513bdb4a6b19b5f800000 (contracts/Power.sol#314)
```



- No major issues were found by Slither. All the reentrancies flagged by Slither are false positives. The functions ProductTokenHighBase.buy(), ProductTokenHighBase.sell() and ProductTokenHighBase.tradein() are all protected by the nonReentrant modifier.

4.2 AUTOMATED SECURITY SCAN

Description:

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

MythX results:

BancorBondingCurve.sol

Report for contracts/BancorBondingCurve.sol
https://dashboard.mythx.io/#/console/analyses/0c4fc1bf-57d5-45b0-9022-a63d7df71a41

Line	SWC Title	Severity	Short Description
3	(SWC-103) Floating Pragma	Low	A floating pragma is set.

Power.sol

Report for contracts/Power.sol
https://dashboard.mythx.io/#/console/analyses/fd5419aa-a000-401e-a7cf-bc8c4110716a

Line	SWC Title	Severity	Short Description
3	(SWC-103) Floating Pragma	Low	A floating pragma is set.

ProductNft.sol

Report for contracts/ProductNft.sol
https://dashboard.mythx.io/#/console/analyses/88fb232b-24ee-46b8-b1fa-fa047550c287

Line	SWC Title	Severity	Short Description
2	(SWC-103) Floating Pragma	Low	A floating pragma is set.

ProductTokenCore.sol

Report for contracts/ProductTokenCore.sol
https://dashboard.mythx.io/#/console/analyses/a15c5f25-46f0-4238-abf9-acf62355aaef

Line	SWC Title	Severity	Short Description
3	(SWC-103) Floating Pragma	Low	A floating pragma is set.

ProductTokenHighBase.sol

Report for contracts/ProductTokenHighBase.sol
https://dashboard.mythx.io/#/console/analyses/103b3df7-dda8-4a9c-811c-8c98142c6bd7

Line	SWC Title	Severity	Short Description
3	(SWC-103) Floating Pragma	Low	A floating pragma is set.

- No major issues were found by MythX. The floating pragma is a false positive as the pragma is set in the truffle-config.js file to the

0.8.10 version.

THANK YOU FOR CHOOSING

// HALBORN