Dec 19, 2022

ZAN

Security Assessment BitKeep Exchange

Professional Service

Table of Contents

1. Overview

1.1. Executive Summary

The BitKeep Exchange library contains a set of smart contracts for EVM-based blockchains (Ethereum, BNB Chain, etc.), which serves as a critical part of the BitKeep Exchange protocol. BitKeep Exchange allows users to batch buy NFTs from different marketplaces using ethers as well as ERC20 tokens.The security assessment was scoped to all the source code of the project as well as any contract dependencies that were not part of an officially recognized library.
We performed a comprehensive examination in combination of Static Analysis, Formal Verification and Manual Review techniques. In our review of the contract, 2 high, 1 medium and 9 informational issues were identified. The project team addressed all issues identified in the initial assessment.

1.2. Project Summary

Project Name	BitKeep Exchange
Platform	Ethereum, BNB Chain, Polygon, Optimism, Arbitrum
Language	Solidity
Code Repository	https://github.com/bitkeepwallet/bkexchange
Commit	ebc8de83aea4ade060193277c8d92edc19b50952

1.3. Assessment Summary

Delivery Date	Dec. 19th, 2022
Audit Methodology	Static Analysis, Formal Verification, Manual Review

1.4. Assessment Scope

ID	File
01	contracts/BKExchangePeriphery.sol
02	contracts/BKExchangeRouter.sol
03	contracts/MarketRegistry.sol
04	contracts/BKCommon.sol
05	contracts/utils/TransferHelper.sol
06	contracts/lib/ConsiderationStructs.sol
07	contracts/lib/ConsiderationEnums.sol
08	contracts/market/SeaportMarket.sol
09	contracts/interfaces/IBKCommon.sol
10	contracts/interfaces/ISeaportMarket.sol
11	contracts/interfaces/IBKErrors.sol

2. Checklist

2.1. General Vulnerability

Reentrancy	DelegateCall
Integer Overflow	Input Validation
Unchecked this.call	Frozen Money
Arbitrary External Call	Unchecked Owner Transfer
Do-while Continue	Right-To-Left-Override Character
Unauthenticated Storage Access	Risk For Weak Randomness
TxOrigin	Missing Checks for Return Values
Diamond Inheritance	ThisBalance
VarType Deduction	Array Length Manipulation
Uninitialized Variable	Shadow Variable
Divide Before Multiply	Function Not Working

2.2. Code Conventions

Compiler Version	Improper State Variable Modification
Function Visibility	Deprecated Function
Externally Controlled Variables	Code Style
Constant Specific	Event Specific
Return Value Unspecified	Nonexistent Error Message
Reference Variable Specification	Import Issue
Compare With Timestamp/Block Number/Blockhash	Constructor in Base Contract Not Implemented
Delete Struct Containing the Mapping Type	Usage of '=+'
Paths in the Modifier Not End with "_" or Revert	Non-payable Public Functions Use msg.value
SafeMath Issue	Compiler Error/Warning
ERC20/ERC721/ERC1155 Standard Specification	Anti-reentry Lock Specific
Nested Function Calls	Inheritance Issue
Signature Replay Risk	Missing Event

2.3. Gas Optimization

Tautology Issue	Loop Depends on Array Length
Redundant/Duplicated/Dead Code	Code Complexity/Code Inefficiency
Undeclared Resource	Optimizable Return Statement
Unused Resource	Duplicate Code

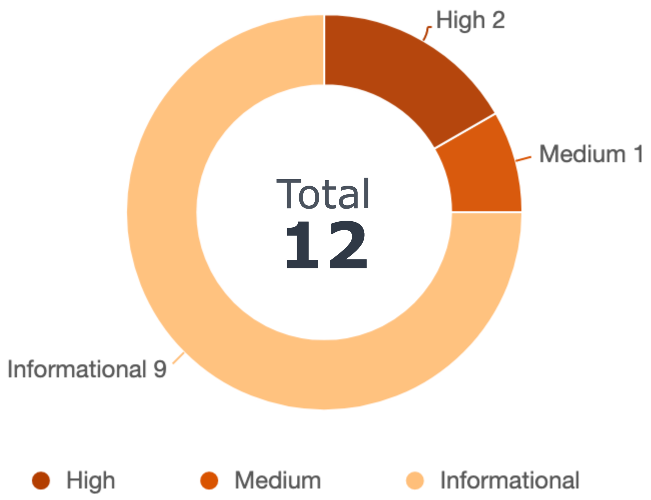
2.4. Compiler Bug

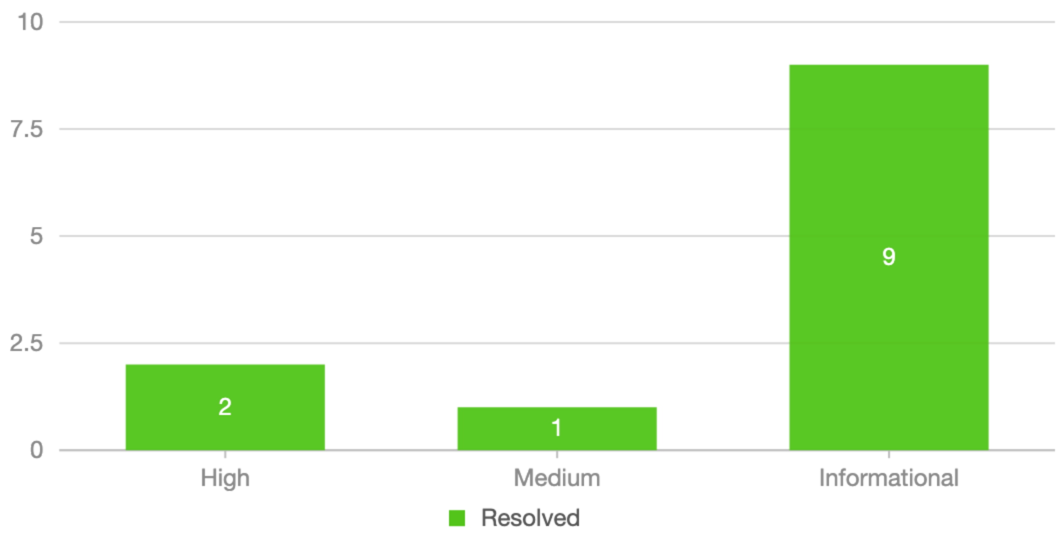
Affected by Compiler Bug

2.5. Logical Issue

The Code Implementation is Consistent With Comments, Project White Papers and Other Materials
Permission Check
Address Check

3. Findings





ID	Title	Category	Severity	Status
H-01	Missing onERC721Received and onERC1155Received Implementation	Logical Issue	● High	Resolved
H-02	Incorrect Transfer Token to BKExchangeRouter	Logical Issue	● High	Resolved

ID	Title	Category	Severity	Status
M-03	Invalid _revertIfTrxFails Flag	Code Conventions	● Medium	Resolved
I-04	Warning of Potential Conflict Storage	Logical Issue	● Informational	Resolved
I-05	Length not checked	Logical Issue	● Informational	Resolved
I-06	Not Check If Market Id Exists	Logical Issue	● Informational	Resolved
I-07	Incorrect Use of ApproveMax	Logical Issue	● Informational	Resolved
I-08	Bad Use of Modifier handleDustXXX	Code Conventions	● Informational	Resolved
I-09	Not Inherited Interface	Logical Issue	● Informational	Resolved
I-10	Incompatible Interfaces and Implementations	Logical Issue	● Informational	Resolved
I-11	Redundant Code	Gas Optimization	● Informational	Resolved
I-12	Use calldata Instead of memory	Gas Optimization	● Informational	Resolved

H-01 | Missing onERC721Received and onERC1155Received Implementation

High : Logical Issue

File Location : contracts/BKExchangePeriphery.sol

Description

BKExchangePeriphery will interact with seaport by delegate call to SeaportMarket, which is about to receive ERC721 and ERC1155 tokens. onERC721Received or onERC1155Received interfaces must be implemented by contracts if they want to accept tokens through safeTransferFrom.

Recommendation

We recommend to inherit OpenZeppelin IERC721Receiver and IERC1155Receiver, and implement onERC721Received or onERC1155Received interfaces. It is also recommended to fully test contracts before audit.

Alleviation

The project team deleted the buyByFulfillBasicOrder() function, therefore the BKExchangePeriphery no longer needs to receive NFT. The issue was resolved in commit 7185acaba8352fc4c5987f7bc569b922d4d57841.

H-02 | Incorrect Transfer Token to BKExchangeRouter

High : Logical Issue

File Location : contracts/market/SeaportMarket.sol:128-142

Description

In function _buyByFulfillBasicOrder, token is transferred back to BKExchangeRouter after calling seaport fulfillBasicOrder. As the following code snippet shows, the msg.sender is BKExchangeRouter. This will fail as BKExchangeRouter not implement onERC721Received nor onERC1155Received interfaces.

```
128  if(_isERC721) {
129      IERC721(fulfillBasicOrderBuy.basicOrderParameters.offerToken).safeTransferFrom(
130              address(this),
131              msg.sender,
132              fulfillBasicOrderBuy.basicOrderParameters.offerIdentifier
133      );
134  } else {
135      IERC1155(fulfillBasicOrderBuy.basicOrderParameters.offerToken).safeTransferFrom(
136              address(this),
137              msg.sender,
138              fulfillBasicOrderBuy.basicOrderParameters.offerIdentifier,
139              fulfillBasicOrderBuy.basicOrderParameters.offerAmount,
140              "0x"
141      );
142  }
```

Recommendation

We suggest to add a receiver address in struct FulfillBasicOrderBuy to transfer the token back, instead of through BKExchangeRouter, as the router should not receive any token during the trade.

Alleviation

The project team deleted the buyByFulfillBasicOrder() function. The issue was resolved in commit 7185acaba8352fc4c5987f7bc569b922d4d57841.

M-03 | Invalid _revertIfTrxFails Flag

Medium : Code Conventions

File Location : contracts/market/SeaportMarket.sol:108-143

Description

In the function _buyByFulfillBasicOrder, if the call to the function in the contract SEAPORT1_1 fails, in other words, the variable success on line 120 is false, then the ERC721 token or ERC1155 token should not be transferred. But in the current implementation, if both success and _revertIfTrxFails are false, ERC721 or ERC1155 token transfer will still be performed and resulting in revert of whole transaction. Thus the transaction will always revert no matter how _revertIfTrxFails flag is set.

```
108  function _buyByFulfillBasicOrder(
109      FulfillBasicOrderBuy calldata fulfillBasicOrderBuy,
110      bool _isERC721,
111      bool _revertIfTrxFails
112  ) internal {
113      bytes memory _data = abi.encodeWithSelector(
114          ISeaport.fulfillBasicOrder.selector,
115          fulfillBasicOrderBuy.basicOrderParameters
116      );
117
118      (bool success, ) = SEAPORT1_1.call{value:
     fulfillBasicOrderBuy.currentPrice}(_data);
119
120      if (!success && _revertIfTrxFails) {
121          // Copy revert reason from call
122          assembly {
123              returndatacopy(0, 0, returndatasize())
124              revert(0, returndatasize())
125          }
126      }
127
128      if(_isERC721) {
129
     IERC721(fulfillBasicOrderBuy.basicOrderParameters.offerToken).safeTransferFrom(
130              address(this),
131              msg.sender,
132              fulfillBasicOrderBuy.basicOrderParameters.offerIdentifier
133          );
134      } else {
135
     IERC1155(fulfillBasicOrderBuy.basicOrderParameters.offerToken).safeTransferFrom(
136              address(this),
137              msg.sender,
138              fulfillBasicOrderBuy.basicOrderParameters.offerIdentifier,
139              fulfillBasicOrderBuy.basicOrderParameters.offerAmount,
140              "0x"
141          );
142      }
143  }
```

Recommendation

It is recommended to judge the variables success and _revertIfTrxFails separately as below.

```
if (success) {
    if(_isERC721) {
IERC721(fulfillBasicOrderBuy.basicOrderParameters.offerToken).safeTransferFrom(
          address(this),
          msg.sender,
          fulfillBasicOrderBuy.basicOrderParameters.offerIdentifier
      );
    } else {
IERC1155(fulfillBasicOrderBuy.basicOrderParameters.offerToken).safeTransferFrom(
          address(this),
          msg.sender,
          fulfillBasicOrderBuy.basicOrderParameters.offerIdentifier,
          fulfillBasicOrderBuy.basicOrderParameters.offerAmount,
          "0x"
      );
    }
} else if (_revertIfTrxFails) {
    // Copy revert reason from call
    assembly {
        returndatacopy(0, 0, returndatasize())
        revert(0, returndatasize())
    }
}
```

Alleviation

The project team deleted the buyByFulfillBasicOrder() function. The issue was resolved in commit 7185acaba8352fc4c5987f7bc569b922d4d57841.

I-04 | Warning of Potential Conflict Storage

Informational : Logical Issue

File Location : contracts/BKExchangePeriphery.sol,contracts/MarketRegistry.sol

Description

If _isLib is ture, the function will execute _proxy.delegatecall(xxx). The BKExchangePeriphery contract has already taken the first 4 slot. Here may have storage conflict between the BKExchangePeriphery contract and the _proxy contract, if the _isLib flag do not indicate a library contract.

```
function _trade(
    TradeDetails[] memory _tradeDetails,
    address _userAddr,
    bool _requireAllSuccess
) internal {
    for (uint256 i = 0; i < _tradeDetails.length; i++) {
        (address _proxy, bool _isLib, bool _isActive) =
marketRegistry.markets(_tradeDetails[i].marketId);
        require(_isActive, "_trade: InActive Market");

        (bool success, bytes data) = _isLib
            ? _proxy.delegatecall(_tradeDetails[i].tradeData)
            : _proxy.call{value:_tradeDetails[i].value}
(_tradeDetails[i].tradeData);

        if(_requireAllSuccess) _checkCallResult(success);
        if(!success){
            emit TradeError(_userAddr, i, _tradeDetails[i], data);
        }
    }
}
```

Storage layout of BKExchangePeriphery

```
+------------------------------------+--------------------------+------+--------+
|                Name                |           Type           | Slot | Offset|
+------------------------------------+--------------------------+------+--------+
|            Ownable._owner          |         address          |  0   |   0    |
|           Pausable._paused         |          bool            |  0   |   20   |
|      ReentrancyGuard._status       |         uint256          |  1   |   0    |
|        BKCommon.isOperator         | mapping(address => bool) |  2   |   0    |
|     BKExchangePeriphery.bkswap     |         address          |  3   |   0    |
| BKExchangePeriphery.openForTrades  |          bool            |  3   |   20   |
| BKExchangePeriphery.marketRegistry|      MarketRegistry      |  4   |   0    |
+------------------------------------+--------------------------+------+--------+
```

Recommendation

Please make sure there is no storage conflict between the BKExchangePeriphery contract and the _proxy contract, aka, make sure the _isLib flag do indicate a library contract.

Alleviation

The project team is aware of this issue and will be careful in subsequent development.

I-05 | Length not checked

Informational : Logical Issue

File Location : contracts/BKExchangeRouter.sol:15, contracts/BKCommon.sol:63, contracts/MarketRegistry.sol: 20

Description

Whether or not the length of _tokenIns and _amountIns are same is not checked in BKExchangeRouter.sol.
Whether or not the length of ids and amounts are same is not checked in BKCommon.sol.
Whether or not the length of proxies and isLibs are same is not checked in MarketRegistry.sol.

contracts/BKExchangeRouter.sol

```
function runWithERC20s(
address[] calldata _tokenIns, uint256[] calldata _amountIns, bytes calldata
_data
)
external
payable
whenNotPaused
nonReentrant
{
    for (uint256 i = 0; i < _tokenIns.length; i++) {
        TransferHelper.safeTransferFrom(
            _tokenIns[i],
            msg.sender,
            BK_EXCHANGE,
            _amountIns[i]
        );
    }

    (bool success, bytes memory resultData) = BK_EXCHANGE.call{
        value : msg.value
    }(_data);

    if (!success) {
        _revertWithData(resultData);
    }
}
```

contracts/BKCommon.sol

```
function rescueERC1155(
address asset, uint256[] calldata ids, uint256[] calldata amounts, address
recipient
) onlyOwner external
{
    for (uint256 i = 0; i < ids.length; i++) {
        IERC1155(asset).safeTransferFrom(address(this), recipient, ids[i],
amounts[i], "");
    }
    emit RescueERC1155(asset, recipient, ids, amounts);
}
```

contracts/MarketRegistry.sol

```
constructor(address[] memory proxies, bool[] memory isLibs, address _owner) {
    for (uint256 i = 0; i < proxies.length; i++) {
        markets.push(Market(proxies[i], isLibs[i], true));
        emit SetMarketProxy(i, Market(proxies[i], isLibs[i], true));
    }
    _transferOwnership(_owner);
}
```

Recommendation

We recommend to add length check in above functions or front-end application.

Alleviation

The project team added length check to above functions. The issue was resolved in commit 7185acaba8352fc4c5987f7bc569b922d4d57841.

I-06 | Not Check If Market Id Exists

Informational : Logical Issue

File Location : contracts/BKExchangePeriphery.sol:118

Description

In the function _trade, when obtaining market information through the marketId stored in the parameter _tradeDetails, if the marketId exceeds the length of marketRegistry.markets, the transaction will be directly reverted without a clear revert reason.

```
112  function _trade(
113      TradeDetails[] memory _tradeDetails,
114      address _userAddr,
115      bool _requireAllSuccess
116  ) internal {
117      for (uint256 i = 0; i < _tradeDetails.length; i++) {
118          (address _proxy, bool _isLib, bool _isActive) =
     marketRegistry.markets(_tradeDetails[i].marketId);
119          require(_isActive, "_trade: InActive Market");
120
121          (bool success, bytes data) = _isLib
122              ? _proxy.delegatecall(_tradeDetails[i].tradeData)
123              : _proxy.call{value:_tradeDetails[i].value}
     (_tradeDetails[i].tradeData);
124
125          if(_requireAllSuccess) _checkCallResult(success);
126          if(!success){
127              emit TradeError(_userAddr, i, _tradeDetails[i], data);
128          }
129      }
130  }
```

Recommendation

We recommend to add length check in above function or front-end application.

Alleviation

The project team added marketId check. The issue was resolved in commit 7185acaba8352fc4c5987f7bc569b922d4d57841.

I-07 | Incorrect Use of ApproveMax

Informational : Logical Issue

File Location : contracts/BKExchangePeriphery.sol:136

Description

Set the 3rd parameter of TransferHelper.approveMax as type(uint256).max will constantly call safeApprove, which violates the purpose of TransferHelper.approveMax to only approve once to save gas.

contracts/BKExchangePeriphery.sol

```
function _approveToSwap(
    address[] calldata _allTokens
) internal {
    for (uint256 i = 0; i < _allTokens.length; i++) {
        TransferHelper.approveMax(_allTokens[i], bkswap, type(uint256).max);
    }
}
```

contracts/utils/TransferHelper.sol

```
function approveMax(
    IERC20 _token,
    address _spender,
    uint256 _amount
) internal {
    uint256 allowance = _token.allowance(address(this), address(_spender));
    if (allowance < _amount) {
        if (allowance > 0) {
            _token.safeApprove(address(_spender), 0);
        }
        _token.safeApprove(address(_spender), type(uint256).max);
    }
}
```

Recommendation

Set the 3rd parameter of TransferHelper.approveMax as actual swap amount of current transaction token amount, just like AggregationFeature.sol. If this value is not accessible, it can be set as type(uint256).max/2.

```
function _approveToSwap(
    address[] calldata _allTokens
) internal {
    for (uint256 i = 0; i < _allTokens.length; i++) {
        TransferHelper.approveMax(IERC20(_allTokens[i]), bkswap, type(uint256).
max/2);
    }
}
```

Alleviation

The project team followed our advice and updated the code in commit 7185acaba8352fc4c5987f7bc569b922d4d57841.

I-08 | Bad Use of Modifier handleDustXXX

Informational : Code Conventions

File Location : contracts/BKExchangePeriphery.sol:58,67

Description

The handleDustETH and handleDustERC20s functions are implemented as modifiers, which is not a common practice. Modifiers are usually used in access control scenarios, or to avoid redundant code, or both. handleDustETH and handleDustERC20s are neither access control nor redundant code, which should not be implemented as modifiers.

```
modifier handleDustETH(address _userAddr) {
    _;

    uint256 newBalance = address(this).balance;
    if(newBalance > 0){
        TransferHelper.safeTransferETH(_userAddr, newBalance);
    }
}

modifier handleDustERC20s(address[] calldata _allTokens, address _userAddr) {
    _;

    uint256 newBalance = address(this).balance;
    if (newBalance > 0) {
        TransferHelper.safeTransferETH(_userAddr, newBalance);
    }

    for (uint256 i = 0; i < _allTokens.length; i++) {
        uint256 erc20NewBalance = IERC20(_allTokens[i]).balanceOf(address(this
));

        if(erc20NewBalance > 0){
            TransferHelper.safeTransfer(
                _allTokens[i],
                _userAddr,
                erc20NewBalance
            );
        }
    }
}
```

Recommendation

Define handleDustETH and handleDustERC20s as internal functions to improve code readability. Remove ETH handling logic in handleDustERC20s to further clarify the code logic.

Alleviation

The project team followed our advice and updated the code in commit 7185acaba8352fc4c5987f7bc569b922d4d57841.

I-09 | Not Inherited Interface

Informational : Logical Issue

File Location : contracts/interfaces/IBKCommon.sol, contracts/BKCommon.sol

Description

The IBKCommon interface are not inherited by the BKCommon contract.IBKCommon.sol

IBKCommon.sol

```
+-------------------------------+------------+
|              Name             |     ID     |
+-------------------------------+------------+
| setOperator(address[],bool)  | 0x1ed6144e |
|           pause()             | 0x8456cb59 |
|          unpause()            | 0x3f4ba83a |
|      rescueETH(address)       | 0x04824e70 |
| rescueERC20(address,address) | 0x5d799f87 |
+-------------------------------+------------+
```

BKCommon.sol

```
+------------------------------------------------------+------------+
|                         Name                         |            |
+------------------------------------------------------+------------+
|             setOperator(address[],bool)              | 0x1ed6144e |
|                       pause()                        | 0x8456cb59 |
|                      unpause()                       | 0x3f4ba83a |
|             rescueERC20(address,address)             | 0x5d799f87 |
|       rescueERC721(address,uint256[],address)        | 0x26e2dca2 | -- missing
| rescueERC1155(address,uint256[],uint256[],address)  | 0xb7ce33a2 | -- missing
|                  rescueETH(address)                  | 0x04824e70 |
|                      receive()                       |            |
+------------------------------------------------------+------------+
```

Recommendation

We recommend to inherit IBKCommon interface in BKCommon contract.

BKCommon.sol

```
import "./interfaces/IBKCommon.sol";

contract BKCommon is IBKCommon, IBKErrors, Ownable, Pausable, ReentrancyGuard {
```

Alleviation

The project team followed our advice and updated the code in commit 7185acaba8352fc4c5987f7bc569b922d4d57841.

I-10 | Incompatible Interfaces and Implementations

Informational : Logical Issue

File Location :

Description

1. Funciton-IDs of buyByFulfillBasicOrder, buyByFulfillAdvancedOrder and buyByFulfillAvailableAdvancedOrders in the ISeaportMarket interface are not the same as SeaportMarket contract. Therefore, any contract calling functions in ISeaportMarket may fail or suffer unexpected behaviors. An use case is showed below. When executing the encoded _data, the buyByFulfillBasicOrder function cannot be found, the execution fails or suffers unexpected behaviors depending on whether or not there is a default fallback() function.

2. Functions fulfillBasicOrder, fulfillAdvancedOrder and fulfillAvailableAdvancedOrders are not implemented anywhere. In addition, those functions are the same with ISeaport interface, which are redundant. We recommend to delete them (refer to the recommendation section).

contracts/market/SeaportMarket.sol

```
bytes memory _data = abi.encodeWithSelector(
    ISeaport.fulfillBasicOrder.selector,
    fulfillBasicOrderBuy.basicOrderParameters
);
```

Function Signature of ISeaportMarket

```
+-----------------------------------------+------------+
|     Name                                |     ID     |
+-----------------------------------------+------------+
| fulfillBasicOrder                       | 0xde869052 | -- missing in contract
| fulfillAdvancedOrder                    | 0x5eb295ef | -- missing in contract
| fulfillAvailableAdvancedOrders          | 0x9a3d9c0b | -- missing in contract
| buyByFulfillBasicOrder                  | 0xabe88584 | -- mismatch
| buyByFulfillAdvancedOrder               | 0x17d6f071 | -- mismatch
| buyByFulfillAvailableAdvancedOrders     | 0x67f2dcab | -- mismatch
+-----------------------------------------+------------+
```

Function Signature of SeaportMarket

```
+------------------------------------------+------------+
|           Name                           |     ID     |
+------------------------------------------+------------+
| buyByFulfillBasicOrder                   | 0x026a04cf |
| buyByFulfillAvailableAdvancedOrders      | 0x91392c2c |
| buyByFulfillAdvancedOrder                | 0x24160d74 |
| rescueETH                                | 0x04824e70 |
| rescueERC20                              | 0x5d799f87 |
| rescueERC721                             | 0x26e2dca2 |
| rescueERC1155                            | 0xb7ce33a2 |
| SEAPORT1_1()                             | 0xaa8a3a25 |
| Owner()                                  | 0xb4a99a4e |
+------------------------------------------+------------+
```

Function Signature of ISeaport

```
+------------------------------------------+------------+
|      Name                                |     ID     |
+------------------------------------------+------------+
| fulfillBasicOrder                        | 0xde869052 |
| fulfillAdvancedOrder                     | 0x5eb295ef |
| fulfillAvailableAdvancedOrders           | 0x9a3d9c0b |
+------------------------------------------+------------+
```

Recommendation

1. Either change interface file or implementation file to make sure the functions in interface and implementation have the same function ids.

2. As the fulfillBasicOrder fulfillAdvancedOrder and fulfillAvailableAdvancedOrders are the same with those in ISeaport, we recommend to delete the redundant functions in ISeaportMarket.

Alleviation

1. The project team deleted the redundant code in ISeaportMarket.

2. The project team deleted several functions in ISeaportMarket. However the functions remained still have different function ids with those in SeaportMarket contract. It is a very tricky situation. The EVM compiler(above 0.8 was tested) treats interface and library differently when calculate function ids(signatures) in the situation that the function parameter(s) contains struct(s). If it is a interface, the compiler will expand the struct, then calculate the function id. However, if it is a library, the compiler will not expand the struct, instead, it will use struct name to calculate the function id. Therefore, it is not possible to achieve the same function id between interface and library in this situation.
The project team was aware of this issue. They manually modified the function ids in the ABI file of ISeaportMarket interface, making them consistent with contract SeaportMarket.

I-11 | Redundant Code

Informational : Gas Optimization

File Location : contracts/market/SeaportMarket.sol:212-242

Description

The following functions in a library contract are useless:

- rescueETH
- rescueERC20
- rescueERC721
- rescueERC1155
- _transferEth

Recommendation

Remove above functions from SeaportMarket.

Alleviation

The project team deleted the redundant code. The issue was resolved in commit 7185acaba8352fc4c5987f7bc569b922d4d57841.

I-12 | Use calldata Instead of memory

Informational : Gas Optimization

File Location : contracts/market/SeaportMarket.sol: 26, 27, 146, 155, contracts/BKExchangePeriphery.sol:112

Description

Parameters of fulfillAdvancedOrder() in ISeaport interface are stored in memory, which will cost more gas.
Parameters of buyByFulfillAvailableAdvancedOrders() and _buyByFulfillAvailableAdvancedOrders() are stored in memory, which will cost more gas.
Parameter TradeDetails of _trade is stored in memory, which will cost more gas.

contracts/market/SeaportMarket.sol

```
25  function fulfillAdvancedOrder(
26      AdvancedOrder memory advancedOrder,
27      CriteriaResolver[] memory criteriaResolvers,
28      bytes32 fulfillerConduitKey,
29      address recipient
30    ) external payable returns (bool fulfilled);
```

contracts/market/SeaportMarket.sol

```
145  function buyByFulfillAvailableAdvancedOrders(
146      FulfillAvailableAdvancedOrdersBuy[] memory
     fulfillAvailableAdvancedOrdersBuys,
147      bool revertIfTrxFails
148  ) public {
149      for(uint i = 0; i < fulfillAvailableAdvancedOrdersBuys.length; i++) {
150
     _buyByFulfillAvailableAdvancedOrders(fulfillAvailableAdvancedOrdersBuys[i],
     revertIfTrxFails);
151      }
152  }
153
154  function _buyByFulfillAvailableAdvancedOrders(
155      FulfillAvailableAdvancedOrdersBuy memory
     fulfillAvailableAdvancedOrdersBuy,
156      bool _revertIfTrxFails
157  ) internal {
```

contracts/BKExchangePeriphery.sol

```
112  function _trade(
113      TradeDetails[] memory _tradeDetails,
114      address _userAddr,
115      bool _requireAllSuccess
116  ) internal {
```

Recommendation

We recommend to use calldata instead of memory to save gas.

Alleviation

The project team changed memory type to calldata type to the issues mentioned above.

4. Disclaimer

No description, statement, recommendation or conclusion in this report shall be construed as endorsement, affirmation or confirmation of the project. The security assessment is limited to the scope of work as stipulated in the Statement of Work.

This report is prepared in response to source code, and based on the attacks and vulnerabilities in the source code that already existed or occurred before the date of this report, excluding any new attacks or vulnerabilities that exist or occur after the date of this report. The security assessment are solely based on the documents and materials provided by the customer, and the customer represents and warrants documents and materials are true, accurate and complete.

CONSULTANT DOES NOT MAKE AND HEREBY DISCLAIMS ANY REPRESENTATIONS OR WARRANTIES OF ANY KIND, WHETHER EXPRESS, IMPLIED, STATUTORY OR OTHERWISE, REGARDING THE SERVICES, DELIVERABLES, OR ANY OTHER MATTER PERTAINING TO THIS REPORT.

CONSULTANT SHALL NOT BE RESPONSIBLE FOR AND HEREBY DISCLAIMS MERCHANTABILITY, FITNESS FOR PURPOSE, TITLE, NON-INFRINGEMENT OR NON-APPROPRIATION OF INTELLECTUAL PROPERTY RIGHTS OF A THIRD PARTY, SATISFACTORY QUALITY, ACCURACY, QUALITY, COMPLETENESS, TIMELINESS, RESPONSIVENESS, OR PRODUCTIVITY OF THE SERVICES OR DELIVERABLES.

CONSULTANT EXCLUDES ANY WARRANTY THAT THE SERVICES AND DELIVERABLES WILL BE UNINTERRUPTED, ERROR FREE, FREE OF SECURITY DEFECTS OR HARMFUL COMPONENTS, REVEAL ALL SECURITY VULNERABILITIES, OR THAT ANY DATA WILL NOT BE LOST OR CORRUPTED.

CONSULTANT SHALL NOT BE RESPONSIBLE FOR (A) ANY REPRESENTATIONS MADE BY ANY PERSON REGARDING THE SUFFICIENCY OR SUITABILITY OF SERVICES AND DELIVERABLES IN ANY ACTUAL APPLICATION, OR (B) WHETHER ANY SUCH USE WOULD VIOLATE OR INFRINGE THE APPLICABLE LAWS, OR (C) REVIEWING THE CUSTOMER MATERIALS FOR ACCURACY.

5. Appendix

5.1 Visibility

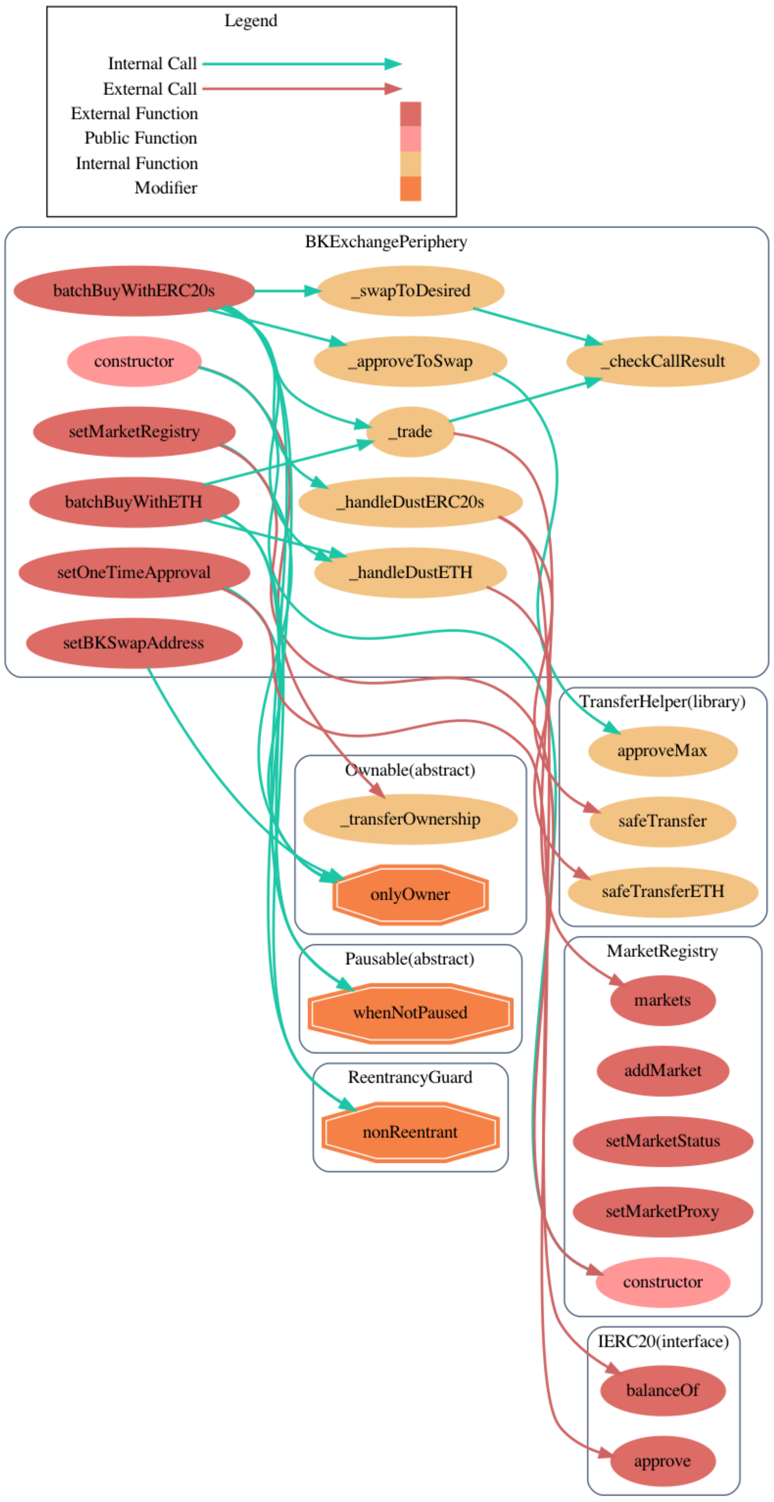
Contract	FuncName	Visibility	Mutability	Modifiers
BKCommon	setOperator	external	Y	onlyOwner
BKCommon	pause	external	Y	onlyOperator
BKCommon	unpause	external	Y	onlyOperator
BKCommon	rescueERC20	external	Y	onlyOperator
BKCommon	rescueERC721	external	Y	onlyOperator
BKCommon	rescueERC1155	external	Y	onlyOperator
BKCommon	rescueETH	external	Y	onlyOperator
BKCommon	_transferEth	internal	Y	
BKCommon	_revertWithData	internal	N	
BKCommon	receive	external	N	
MarketRegistry	_CTOR_	public	Y	
MarketRegistry	marketsLength	public	N	
MarketRegistry	addMarket	external	Y	onlyOwner
MarketRegistry	setMarketStatus	external	Y	onlyOwner
MarketRegistry	setMarketProxy	external	Y	onlyOwner
BKExchangePeriphery	_CTOR_	public	Y	

Contract	FuncName	Visibility	Mutability	Modifiers
BKExchangePeriphery	setBKSwapAddress	external	Y	onlyOwner
BKExchangePeriphery	setMarketRegistry	external	Y	onlyOwner
BKExchangePeriphery	batchBuyWithETH	external	Y	whenNotPaused, nonReentrant
BKExchangePeriphery	batchBuyWithERC20s	external	Y	whenNotPaused, nonReentrant
BKExchangePeriphery	_trade	internal	Y	
BKExchangePeriphery	_approveToSwap	internal	Y	
BKExchangePeriphery	_swapToDesired	internal	Y	
BKExchangePeriphery	_handleDustETH	internal	Y	
BKExchangePeriphery	_handleDustERC20s	internal	Y	
BKExchangePeriphery	_checkCallResult	internal	N	
BKExchangePeriphery	setOneTimeApproval	external	Y	onlyOwner
BKExchangeRouter	_CTOR_	public	Y	
BKExchangeRouter	runWithERC20s	external	Y	whenNotPaused, nonReentrant
BKExchangeRouter	runWithETH	external	Y	whenNotPaused, nonReentrant

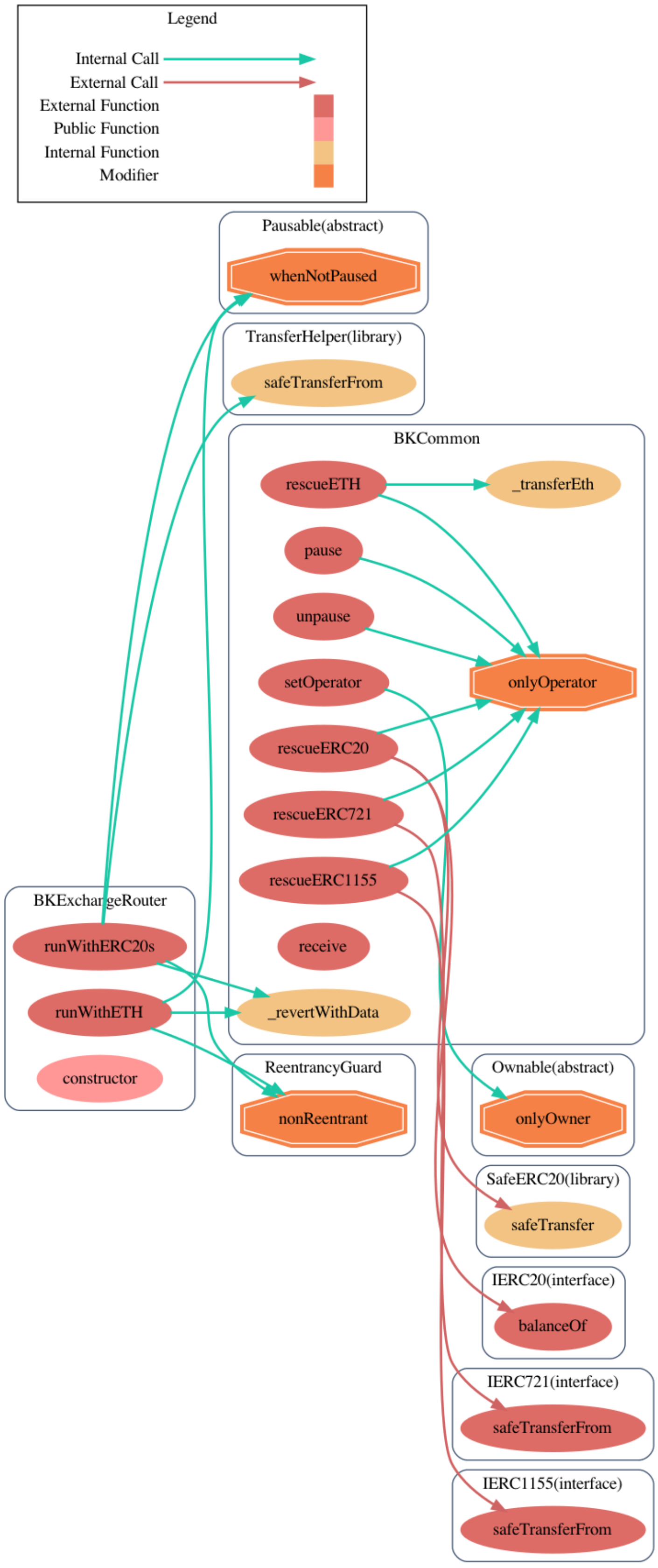
5. Appendix

5.2 Call Graph

contracts/BKExchangePeriphery.sol



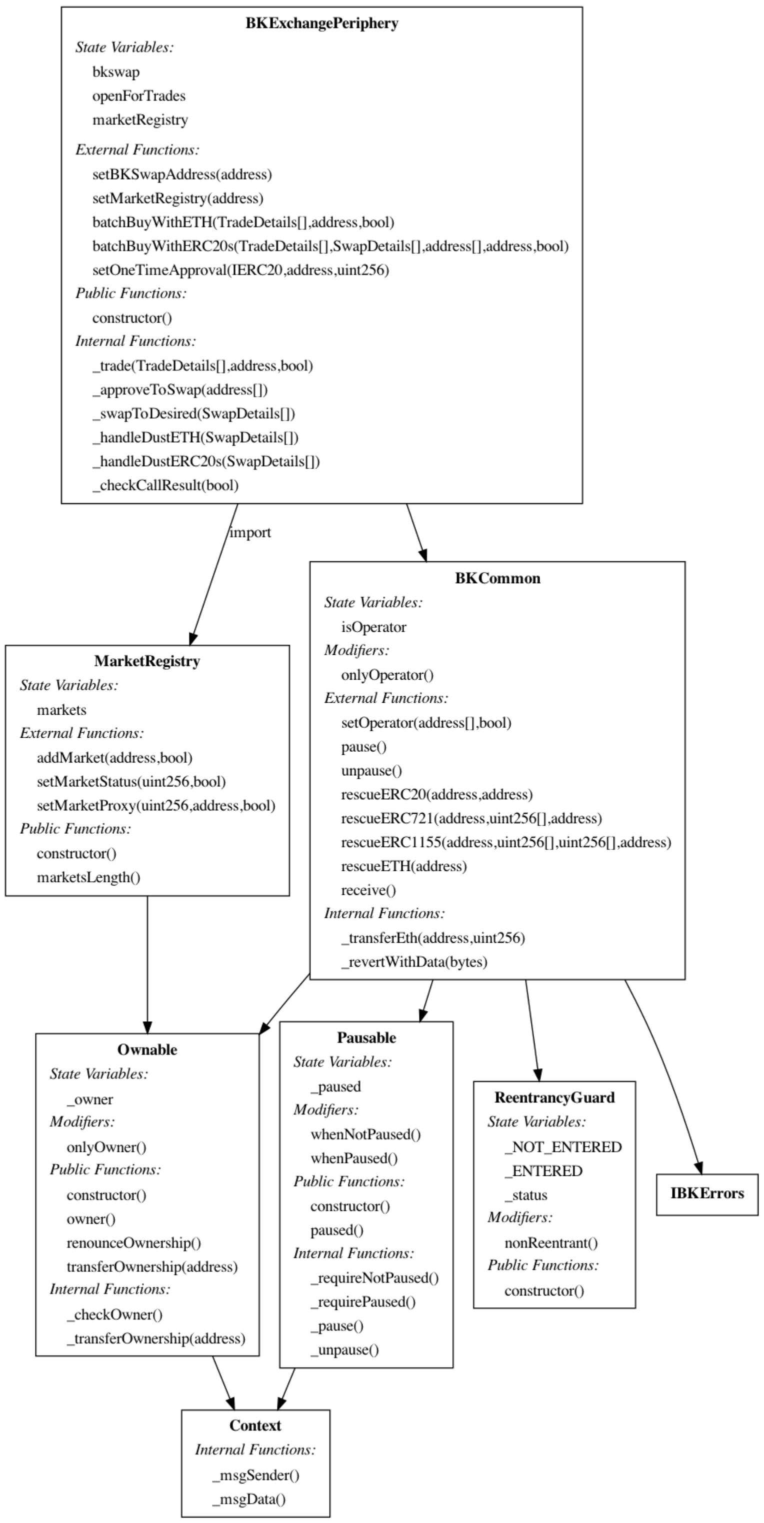
contracts/BKExchangeRouter.sol



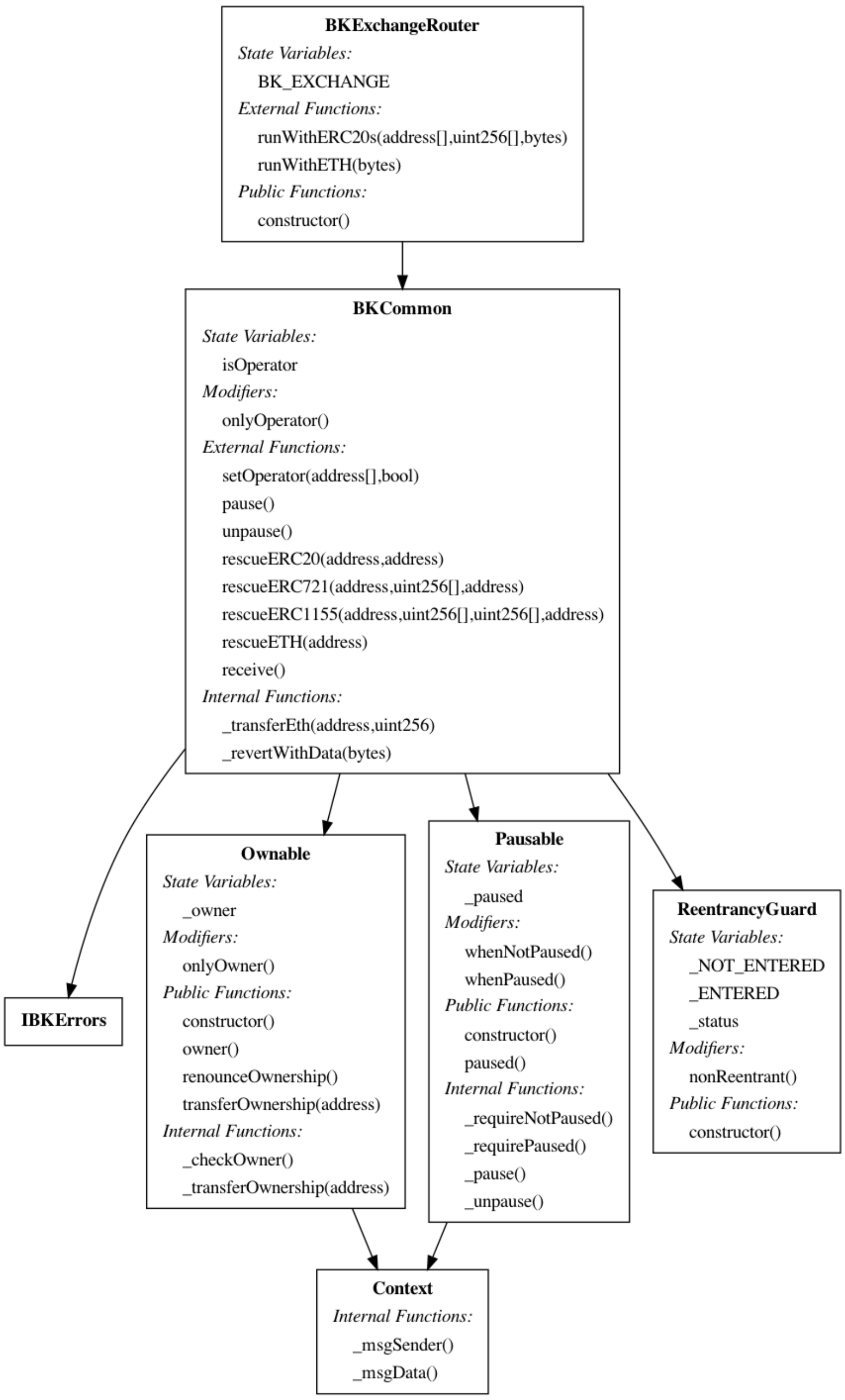
5. Appendix

5.3 Inheritance Graph

contracts/BKExchangePeriphery.sol



contracts/BKExchangeRouter.sol


BKExchangeRouter
State Variables:
BK_EXCHANGE
External Functions:
runWithERC20s(address[],uint256[],bytes)
runWithETH(bytes)
Public Functions:
constructor()
BKCommon
State Variables:
isOperator
Modifiers:
onlyOperator()
External Functions:
setOperator(address[],bool)
pause()
unpause()
rescueERC20(address,address)
rescueERC721(address,uint256[],address)
rescueERC1155(address,uint256[],uint256[],address)
rescueETH(address)
receive()
Internal Functions:
_transferEth(address,uint256)
_revertWithData(bytes)
IBKErrors
Ownable
State Variables:
_owner
Modifiers:
onlyOwner()
Public Functions:
constructor()
owner()
renounceOwnership()
transferOwnership(address)
Internal Functions:
_checkOwner()
_transferOwnership(address)
Pausable
State Variables:
_paused
Modifiers:
whenNotPaused()
whenPaused()
Public Functions:
constructor()
paused()
Internal Functions:
_requireNotPaused()
_requirePaused()
_pause()
_unpause()
ReentrancyGuard
State Variables:
_NOT_ENTERED
_ENTERED
_status
Modifiers:
nonReentrant()
Public Functions:
constructor()
Context
Internal Functions:
_msgSender()
_msgData()