Protocol Audit Report

Prepared by: Bizarro

Table of Contents

Protocol Summary

Our platform processes order executions, position management, and fund movements (fees, profit, and loss) within backend servers. After processing, all relevant events are recorded on-chain through our smart contracts. These contracts act as after-the-fact evidence of backend operations, ensuring transparency and consistency between backend logic and on-chain states.

If discrepancies arise between backend servers and the smart contract logic, state desynchronization may occur, preventing users from performing essential actions (e.g., withdrawals). Therefore, it is imperative that our protocol maintains logical accuracy and consistency between backend processes and smart contracts.

Disclaimer

Bizarro found as many vulnerabilities in the code in the given time period, but holds no responsibilities for the findings provided in this document. A security audit by the team is not an endorsement of the underlying business or product. The audit was time-boxed and the review of the code was solely on the security aspects of the Solidity implementation of the contracts.

Risk Classification

		Impact		
		High	Medium	Low
	High	H	H/M	M
Likelihood	Medium	H/M	M	M/L
	Low	M	M/L	L

We use the CodeHawks severity matrix to determine severity. See the documentation for more details.

Scope

```
Files:
All contracts within /src/** folder,
excluding src/peripherals/Delegation.sol
```

Roles

Executive Summary

Issues found

Severity	**Number of issues found**
High	1
Medium	1
Low	9
Info	0
Gas	0
Total	11

Findings

High

[H-1] `WithdrawlHandler` can have locked eth in the contract.

Description: In `WithdrawHandler` contract there is a payable receive function meaning the contract can receive eth into the contract, and there is no withdraw function in the contract.

Impact: This can lead to locked Eth into the contract.

Recommended Mitigation: remove the payable keyword from the function.

Medium

[M-1] Potential DoS at `Entrypoint::submitTransactions`

Description: In `Entrypoint::submitTransactions` the function takes _transactions as the parameter, which is an array of transactions, an iterator goes through the entire _transactions array and perform the task based on the transaction. If the length of the array is very large then the gas used by the function will be very high and can lead to potential DoS.

Impact: The function `Entrypoint::submitTransactions` won't work if the length of the _transaction array is very large causing DoS.

Proof of Concept:

```
 function submitTransactions(Transaction[] calldata _transactions, uint256
_batchId) external onlyExecutor {
        if (_batchId != nextBatchId) {
            revert Entrypoint_InvalidBatchId();
        }
        // e get the length of transaction
        uint256 _len = _transactions.length;
        // e get the loop integer
        uint256 _i;
        address _handler;
        // q potential DoS
@>        for (_i; _i < _len;) {
            // e get the handler based on the actionType
            _handler = handlers[uint8(_transactions[_i].actionType)];
            // e check if handler address is zero
            if (_handler == address(0)) {
                revert Entrypoint_HandlerIsNotSet();
            }
            // sanity check, txId should goes forward only, ignoring
sequence number as we trust the executor
            if (_transactions[_i].id <= latestTxId) {
                revert Entrypoint_InvalidTxId();
            }
            // e execute action
            IHandler(_handler).executeAction(_transactions[_i].data);
            // e set the latestTxId as the .id
            latestTxId = _transactions[_i].id;

            unchecked {
                ++_i;
            }
        }
        unchecked {
            ++nextBatchId;
        }
    }
```

Recommended Mitigation: check the length of the array beforehand and make sure that the length of the array should not exceed the limit.

Low

[L-1] The return value of the external calls should be checked in `TradeHandler::initialize`.

Description: In `TradeHandler::initialize` the function makes external calls for sanity checks, here the return values from the external checks should be checked.

Proof of Concept:

```
function initialize(address _entrypoint, address _assetService, address
_perpService, address _settlementToken)
        external
        initializer
    {
        __initialize(_entrypoint);

        // sanity check
        // q shouldn't it check if the return is correct or not ?
@>        IAssetService(_assetService).BPS();
@>        IPerpService(_perpService).BPS();
@>
IAssetService(_assetService).collateralTokenDecimals(_settlementToken);

        assetService = _assetService;
        perpService = _perpService;
        settlementToken = _settlementToken;
    }
```

[L-2] Emit an event when their is state change in the contract.

Proof of Concept:

```
    function _initSetPosition(bytes32 _subaccount, uint32 _marketId,
Position calldata _newPosition)
        internal
        onlyOwner
    {
        positions[_subaccount][_marketId] = _newPosition;
@>          _subaccountActivePositions[_subaccount].add(_marketId);

        }
```

[L-3]: Solmate's SafeTransferLib does not check for token contract's existence

There is a subtle difference between the implementation of solmate's SafeTransferLib and OZ's SafeERC20: OZ's SafeERC20 checks if the token is a contract or not, solmate's SafeTransferLib does not. https://github.com/transmissions11/solmate/blob/main/src/utils/SafeTransferLib.sol#L9 `@dev Note that none of the functions in this library check that a token has code at all! That responsibility is delegated to the caller`

▶ 1 Found Instances

- Found in src/peripherals/Vault.sol Line: 6

  ```
  import { SafeTransferLib } from "solmate/utils/SafeTransferLib.sol";
  ```

[L-4]: Event is missing `indexed` fields

Index event fields make the field more quickly accessible to off-chain tools that parse events. However, note that each index field costs extra gas during emission, so it's not necessarily best to index the maximum allowed per event (three fields). Each event should use three indexed fields if there are three or more fields, and gas usage is not particularly of concern for the events in question. If there are fewer than three fields, all of the fields should be indexed.

[L-5]: Empty Block

Consider removing empty blocks.

▶ 1 Found Instances

- Found in src/handlers/BaseHandler.sol Line: 34

  ```
      function _executeAction(bytes memory _data) internal virtual { }
  ```

[L-6]: Large literal values multiples of 10000 can be replaced with scientific notation

Use `e` notation, for example: `1e18`, instead of its full numeric value.

▶ 1 Found Instances

- Found in src/handlers/BaseHandler.sol Line: 9

```
	uint256 public constant BPS = 10000;
```

[L-7]: Unused Custom Error

it is recommended that the definition be removed when custom error is unused

▶ 7 Found Instances

- Found in src/handlers/CollateralSeizeHandler.sol Line: 31

```
	error CollateralSeizeHandler_InvalidAddress();
```

- Found in src/handlers/TradeHandler.sol Line: 65

```
	error TradeHandler_OrderFilled();
```

- Found in src/interfaces/IEntrypoint.sol Line: 6

```
	error Entrypoint_InvalidNonce();
```

- Found in src/interfaces/IEntrypoint.sol Line: 8

```
	error Entrypoint_InvalidSignature();
```

- Found in src/interfaces/services/IPerpService.sol Line: 46

```
	error PerpService_InvalidAddress();
```

- Found in src/interfaces/services/IPerpService.sol Line: 47

```
	error PerpService_InvalidMarket();
```

- Found in src/interfaces/services/IPerpService.sol Line: 48

```
	error PerpService_Unauthorized();
```

[L-8]: Uninitialized local variables.

Initialize all the variables. If a variable is meant to be initialized to zero, explicitly set it to zero to improve code readability.

▶ 4 Found Instances

- Found in src/Entrypoint.sol Line: 55

  ```
          uint256 _i;
  ```

- Found in src/services/AssetService.sol Line: 213

  ```
          uint256 _i;
  ```

- Found in src/services/PerpService.sol Line: 163

  ```
          uint256 _i;
  ```

- Found in src/services/PerpService.sol Line: 192

  ```
          uint256 _i;
  ```

[L-9] Unsafe downcast

▶ 1 Found Instances

- Found in src/services/OracleService.sol Line: 80

  ```
      _priceE18 = uint32(uint256((_pricesSlot >> (256 - (32 *
  (_slotOffset + 1))))));
  ```