tags: `Final Report`

Fountain Protocol Incremental Audit (FPIA-1)

April 12, 2022
by **Verilog Audit**

This report presents Verilog's incremental smart contract auditing engagement with Fountain Protocol, especially for its `LPOracleAnchoredView.sol` smart contract. Fountain Protocol is one of the first Lending protocols on the Emerald Paratime of Oasis Network.

Table of Content

- Fountain Protocol Incremental Audit (FPIA-1)
  - Table of Content
  - Summary of the Incremental Audit
  - Privileged Roles
  - Findings & Improvement Suggestions
    - Critical
    - Major
    - Medium

- Minor
- Informational

Summary of the Incremental Audit

We audited Fountain Protocol and the previous auditing report is here (https://hackmd.io/C_lPwlT0TsuONfUWkQzfpw) with hash cc16318c2db70fdc8fbfb52c26c1f7b9d15875f8 (https://github.com/dev-fountain/fountain-protocol/tree/cc16318c2db70fdc8fbfb52c26c1f7b9d15875f8).

This is the incremental audit for file `LPOracleAnchoredView.sol` (https://github.com/dev-fountain/fountain-protocol/blob/e2c39e77c4df4a93f807abdd546a40ff76a7b483/contracts/LPOracleAnchoredView.sol). The main functionality added is the calculation for LP token price.

Privileged Roles

1. The caller of the `constructor` (*i.e.*, the deployer of the smart contract) has the privileged role to select the token pairs and pass the tokens' symbols and addresses to the function.

```
constructor(address _ref,OracleTokenConfig[] memory configs) public {
   ref = IStdReference(_ref);
   for(uint i = 0; i < configs.length; i++){
       OracleTokenConfig memory config = configs[i];
       require(config.baseUnit > 0, "baseUnit must be greater than zero");
       CTokenConfigs[config.symbol] = config;
       cTokenSymbol[config.cToken] = config.symbol;
   }
}

```

Findings & Improvement Suggestions

Informational   Minor   Medium   Major   Critical

	Total	Acknowledged	Resolved
Critical	0	0	0
Major	1	1	1
Medium	0	0	0
Minor	0	0	0
Informational	3	3	3

Critical

none ;)

Major

1. The token decimal alignment is not needed in function `reserveProductAndTotalSupply()` (https://github.com/dev-fountain/fountain-protocol/blob/e2c39e77c4df4a93f807abdd546a40ff76a7b483/contracts/LPOracleAnchoredView.sol#L91). However, the product of prices of the tokens with different decimals should be considered in function `priceProduct()` (https://github.com/dev-fountain/fountain-protocol/blob/e2c39e77c4df4a93f807abdd546a40ff76a7b483/contracts/LPOracleAnchoredView.sol#L103). **major**
   Description: In the AMM model, to determine the product of the amounts of two tokens in a pair, we do not need to cast the decimal to 18. Besides, doing this kind of casting may result in the loss of accuracy if one of the tokens in a pair has the decimal greater than 18.

```
function reserveProductAndTotalSupply(string memory symbol) internal view re
    OracleTokenConfig memory config = CTokenConfigs[symbol];
    IDexPair dexPair = IDexPair(config.underlying);
    totalSUpply = dexPair.totalSupply();
    (uint112 reserve0, uint112 reserve1,) = dexPair.getReserves();
    uint decimal0 = OracleERC20(dexPair.token0()).decimals();
    uint decimal1 = OracleERC20(dexPair.token1()).decimals();
    uint amount0 = uint(reserve0).mul(1e18).div(10 ** decimal0);
    uint amount1 = uint(reserve1).mul(1e18).div(10 ** decimal1);
    product = amount0.mul(amount1);
}
```

Recommendation: We suggest to remove the decimal alignment or do not use this contract for token with decimals greater than 18 otherwise there can be loss of accuracy

Result: Fixed in commit dd9475ebc63c5fbbb396c4c01fbfdb59d8821896 (https://github.com/dev-fountain/fountain-protocol/commit/dd9475ebc63c5fbbb396c4c01fbfdb59d8821896).

Medium

none ;)

Minor

none ;)

Informational

1. Typo in function return values **Informational**
   Description: There is a typo in the return values of function `reserveProductAndTotalSupply()` (https://github.com/dev-fountain/fountain-protocol/blob/e2c39e77c4df4a93f807abdd546a40ff76a7b483/contracts/LPOracleAnchoredView.sol#L91). The `totalSUpply` should be spelled like `totalSupply`. The variables should be spelled in *camel* format.
   Recommendation: `totalSUpply -> totalSupply`.
   Result: Fixed in commit 11f434ccfa13f9ea49d05259c4b0f5e411322aa6 (https://github.com/dev-fountain/fountain-protocol/commit/11f434ccfa13f9ea49d05259c4b0f5e411322aa6).

2. Magic Numbers
   Description: There are some *magic numbers* in the code deck. For example, `1e28` in Line 67 (https://github.com/dev-fountain/fountain-protocol/blob/e2c39e77c4df4a93f807abdd546a40ff76a7b483/contracts/LPOracleAnchoredView.sol#L67), `1e10` in Line 82 (https://github.com/dev-fountain/fountain-protocol/blob/e2c39e77c4df4a93f807abdd546a40ff76a7b483/contracts/LPOracleAnchoredView.sol#L82) and Line 87 (https://github.com/dev-fountain/fountain-protocol/blob/e2c39e77c4df4a93f807abdd546a40ff76a7b483/contracts/LPOracleAnchoredView.sol#L87).
   Informational
   Recommendation: Make these magic numbers constant values with comments.
   Result: Improved in commit bce7296eedc2922fa6ea0ab42a0d718b3ee1ef31 (https://github.com/dev-fountain/fountain-protocol/commit/bce7296eedc2922fa6ea0ab42a0d718b3ee1ef31) and d441a1b0561caf2fbf3065c8266df80381904ac6 (https://github.com/dev-fountain/fountain-protocol/commit/d441a1b0561caf2fbf3065c8266df80381904ac6).

3. Unnecessary ordering between `tokenA` and `tokenB` in function `priceProduct()` (https://github.com/dev-fountain/fountain-protocol/blob/e2c39e77c4df4a93f807abdd546a40ff76a7b483/contracts/LPOracleAnchoredView.sol#L103).

   Informational

   Description: In this function, it purposely changes the order of `tokenA` and `tokenB` and saves the symbol of the token that has a smaller token address in variable `symbol0` and the other in `symbol1` . In fact, the orders of the tokens does not change the result of product (`product = price0.mul(price1)`).

```
function priceProduct(string memory symbol) internal view returns(uint produ
    OracleTokenConfig memory config = CTokenConfigs[symbol];
    string memory symbol0;
    string memory symbol1;
    if(config.tokenA < config.tokenB){
        symbol0 = config.symbolA;
        symbol1 = config.symbolB;
    }else{
        symbol0 = config.symbolB;
        symbol1 = config.symbolA;
    }
    uint price0 = oraclePrice(symbol0).rate;
    uint price1 = oraclePrice(symbol1).rate;
    product = price0.mul(price1);
}
```

Recommendation: Token symbols can be assigned directly without checking the token orders.
Result: Revised in commit ffc99f59d054e78701de8a0a8899faa0f4c33326 (https://github.com/dev-fountain/fountain-protocol/commit/ffc99f59d054e78701de8a0a8899faa0f4c33326).