Kinetic Audit Report

Apr 11, 2024

Table of Contents

Summary

This report has been prepared for Kinetic smart contract, to discover issues and vulnerabilities in the source code of their Smart Contract as well as any contract dependencies that were not part of an officially recognized library. A comprehensive examination has been performed, utilizing Static Analysis and Manual Review techniques.

The auditing process pays special attention to the following considerations:

- Testing the smart contracts against both common and uncommon attack vectors.
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line-by-line manual review of the entire codebase by industry experts.

Overview

Project Summary

Project Name	**Kinetic**
Codebase	**https://github.com/kinetic-market/money-market-contracts**
Commit	**8dc229b2af86eec5ff55742f455904faee451e3e**
Language	**Solidity**

Audit Summary

Delivery Date	**Apr 11, 2024**
Audit Methodology	**Static Analysis, Manual Review**
Total Isssues	**9**

[WP-L1] Incorrect `ftsoRegistry_` productionMode Check

Low

Issue Description

Based on the context, it seems intended to write `require(ftsoRegistry_.productionMode())`, but the current implementation does not check the return value.

See: https://docs.flare.network/apis/smart-contracts/FtsoRegistry/#va_productionmode

https://github.com/kinetic-market/money-market-contracts/blob/0e5b054ae43c349750f79dd14ce005a8e06942ab/contracts/FTSO/ProtocolFTSOOracle.sol#L87-L95

```
87  function setFTSORegistry(IFTSO ftsoRegistry_) external onlyOwner {
88      address ftsoRegistryAddress = address(ftsoRegistry_);
89      require(ftsoRegistryAddress != address(0) && ftsoRegistryAddress !=
    address(this), "invalid FTSO Registry address");
90
91      ftsoRegistry_.productionMode();
92
93      ftsoRegistry = ftsoRegistry_;
94      emit FTSOOracleSet(ftsoRegistryAddress);
95  }
```

Status

Acknowledged

[WP-L2] Using a symbol as the key for `tokenConfigs[token.symbol()]` contradicts best practices.

Low

Issue Description

https://github.com/kinetic-market/money-market-contracts/blob/0e5b054ae43c349750f79dd14ce005a8e06942ab/contracts/FTSO/ProtocolFTSOOracle.sol#L23-L25

```
23  /// @notice Underlying token configs by token symbol
24  mapping(string => TokenConfig) public tokenConfigs;
```

https://github.com/kinetic-market/money-market-contracts/blob/0e5b054ae43c349750f79dd14ce005a8e06942ab/contracts/FTSO/ProtocolFTSOOracle.sol#L50-L67

```
50  function setTokenConfig(TokenConfig memory config) public onlyOwner {
51      if (config.asset == address(0))
52          revert("can't be zero address");
53
54      if (config.maxStalePeriod == 0)
55          revert("max stale period cannot be 0");
56
57      EIP20Interface token = EIP20Interface(config.asset);
58
59      tokenConfigs[token.symbol()] = config;
60
61      emit TokenConfigAdded(
62          config.asset,
63          config.ftsoSymbol,
64          config.maxStalePeriod
65      );
66  }
```

The symbol returned by a token can change, consider using the token's address as the key

instead.

Status

ⓘ Acknowledged

[WP-L3] `getClaimableRewards()` does not account for pending reward accrual since the last update of `rewardIndex` .

Low

Issue Description

https://github.com/kinetic-market/money-market-contracts/blob/0e5b054ae43c349750f79dd14ce005a8e06942ab/contracts/Tokenomics/genesisPools/GenesisPoolStakingContract.sol#L81-L86

```
81  function getClaimableRewards() external view returns(uint) {
82      uint rewardIndexDelta = rewardIndex.sub(supplierRewardIndex[msg.sender]);
83      uint claimableReward =
    rewardIndexDelta.mul(supplyAmount[msg.sender]).div(1e36).add(accruedReward[msg.sender]);
84
85      return claimableReward;
86  }
```

Status

✓ Fixed

[WP-L4] `redeemIndex` may cancel other redemptions unexpectedly.

Low

Issue Description

The redemption corresponding to the same `redeemIndex` is dynamic.

Due to network congestion, users might sign multiple `cancelRedeem` calls repeatedly, inadvertently canceling other redemptions. Alternatively, if the frontend does not refresh in time, users might believe their cancellation attempt failed and retry the transaction with the same parameters, leading to the unintended cancellation of other redemptions.

https://github.com/kinetic-market/money-market-contracts/blob/0e5b054ae43c349750f79dd14ce005a8e06942ab/contracts/Tokenomics/esProtocol.sol#L235-L249

```
235     function cancelRedeem(uint256 redeemIndex) external nonReentrant
    validateRedeem(msg.sender, redeemIndex) {
236       uint256 balance = esProtocolBalances[msg.sender];
237       RedeemInfo storage _redeem = userRedeems[msg.sender][redeemIndex];
238
239       // make redeeming esProtocol available again
240       balance = balance - _redeem.esProtocolAmount;
241       esProtocolBalances[msg.sender] = balance;
242       _transfer(address(this), msg.sender, _redeem.esProtocolAmount);
243
244       emit CancelRedeem(msg.sender, _redeem.esProtocolAmount);
245
246       // remove redeem entry
247       _deleteRedeemEntry(redeemIndex);
248     }
```

https://github.com/kinetic-market/money-market-contracts/blob/0e5b054ae43c349750f79dd14ce005a8e06942ab/contracts/Tokenomics/esProtocol.sol#L289-L292

```
289   function _deleteRedeemEntry(uint256 index) internal {
290     userRedeems[msg.sender][index] =
    userRedeems[msg.sender][userRedeems[msg.sender].length - 1];
291     userRedeems[msg.sender].pop();
292   }
```

Recommendation

Consider adding a new parameter to `cancelRedeem()` :

```
235   function cancelRedeem(uint256 redeemIndex, uint256 redeemsLength) external
    nonReentrant validateRedeem(msg.sender, redeemIndex) {
236     require(userRedeems[msg.sender].length == redeemsLength);
237     uint256 balance = esProtocolBalances[msg.sender];
238     RedeemInfo storage _redeem = userRedeems[msg.sender][redeemIndex];
239
240     // make redeeming esProtocol available again
241     balance = balance - _redeem.esProtocolAmount;
242     esProtocolBalances[msg.sender] = balance;
243     _transfer(address(this), msg.sender, _redeem.esProtocolAmount);
244
245     emit CancelRedeem(msg.sender, _redeem.esProtocolAmount);
246
247     // remove redeem entry
248     _deleteRedeemEntry(redeemIndex);
249   }
```

Status

✓ Fixed

[WP-L5] The existence of `burnRateCalculator` is not checked before using it.

Low

Issue Description

From the context, we infer that `burnRateCalculator` is optional, as the protocol does not mandate its configuration. However, in the current implementation, if it is not set, `getProtocolByVestingDuration()` will unexpectedly revert.

https://github.com/kinetic-market/money-market-contracts/blob/0e5b054ae43c349750f79dd14ce005a8e06942ab/contracts/Tokenomics/esProtocol.sol#L68-L76

```
68    function getProtocolByVestingDuration(uint256 amount, uint256 duration) public
      returns (uint256) {
69        require(duration == minRedeemDuration || duration == maxRedeemDuration,
      "getProtocolByVestingDuration: invalid duration");
70
71        uint256 ratio = duration == maxRedeemDuration ? maxRedeemRatio
72            : (burnRateCalculator.shouldSkipBurnRate(msg.sender, amount) ?
      maxRedeemRatio : minRedeemRatio);
73
74        return amount * ratio / 100;
75    }
```

Recommendation

Consider checking if `burnRateCalculator == address(0)` and using the default (minRedeemRatio) when it is not set.

Status

✓ Fixed

[WP-L6] Price Oracle Cannot Support Tokens with Decimals Greater Than 18

Low

Issue Description

Any token with decimals > 18 will revert at L42 due to underflow.

https://github.com/kinetic-market/money-market-contracts/blob/0e5b054ae43c349750f79dd14ce005a8e06942ab/contracts/OverridablePriceOracle.sol#L33-L49

```
33  function getPrice(address tokenAddress) public view returns (uint price) {
34      EIP20Interface token = EIP20Interface(tokenAddress);
35
36      if (prices[address(token)] != 0) {
37          price = prices[address(token)];
38      } else {
39          price = _getPrice(tokenAddress);
40      }
41
42      uint decimalDelta = uint(18).sub(uint(token.decimals()));
43      // Ensure that we don't multiply the result by 0
44      if (decimalDelta > 0) {
45          return price.mul(10**decimalDelta);
46      } else {
47          return price;
48      }
49  }
```

Status

ⓘ Acknowledged

[WP-I7] `OverridablePriceOracle` prices should potentially have an expiration

Informational

Issue Description

`OverridablePriceOracle` is a centralized and error-prone legacy implementation. Specifically, the owner can set a high-priority override price, but due to the lack of an expiration concept, this price will continue to override the price from FTSO indefinitely. For example, if the owner sets an override price of $1 for USDC, this setting might go unnoticed for a long time. A sudden depeg of USDC could lead to catastrophic outcomes.

We recommend adding an expiration time to the override price at a minimum, after which the price from FTSO should be resumed.

https://github.com/kinetic-market/money-market-contracts/blob/0e5b054ae43c349750f79dd14ce005a8e06942ab/contracts/OverridablePriceOracle.sol#L74-L77

```
74  function setPrice(address asset, uint price) public onlyOwner() {
75      emit PricePosted(asset, prices[asset], price, price);
76      prices[asset] = price;
77  }
```

https://github.com/kinetic-market/money-market-contracts/blob/0e5b054ae43c349750f79dd14ce005a8e06942ab/contracts/OverridablePriceOracle.sol#L33-L49

```
33  function getPrice(address tokenAddress) public view returns (uint price) {
34      EIP20Interface token = EIP20Interface(tokenAddress);
35
36      if (prices[address(token)] != 0) {
37          price = prices[address(token)];
38      } else {
39          price = _getPrice(tokenAddress);
40      }
41
42      uint decimalDelta = uint(18).sub(uint(token.decimals()));
43      // Ensure that we don't multiply the result by 0
```

```
44        if (decimalDelta > 0) {
45            return price.mul(10**decimalDelta);
46        } else {
47            return price;
48        }
49    }
```

Status

ⓘ Acknowledged

[WP-I8] `ProtocolFTSOOracle.getFTSOPrice()` does not support `tokenConfig.ftsoSymbol` with `_assetPriceUsdDecimals > 18` .

Informational

Issue Description

Any token with _assetPriceUsdDecimals > 18 will revert at L113 due to underflow.

https://github.com/kinetic-market/money-market-contracts/blob/0e5b054ae43c349750f79dd14ce005a8e06942ab/contracts/FTSO/ProtocolFTSOOracle.sol#L105-L120

```
105     function getFTSOPrice(TokenConfig memory tokenConfig) internal view returns (uint) {
106         (uint256 _price, uint256 _timestamp, uint256 _assetPriceUsdDecimals) =
107             ftsoRegistry.getCurrentPriceWithDecimals(tokenConfig.ftsoSymbol);
108 
109         require(_price > 0, "invalid price");
110         require(_assetPriceUsdDecimals > 0, 'invalid exponential');
111         require(block.timestamp.sub(_timestamp) <= tokenConfig.maxStalePeriod, "stale price");
112 
113         uint decimalDelta = uint(18).sub(_assetPriceUsdDecimals);
114 
115         if (decimalDelta > 0) {
116             return _price.mul(10**decimalDelta);
117         } else {
118             return _price;
119         }
120     }
```

https://github.com/kinetic-market/money-market-contracts/blob/0e5b054ae43c349750f79dd14ce005a8e06942ab/contracts/SafeMath.sol#L50-L75

```
50     /**
51      * @dev Returns the subtraction of two unsigned integers, reverting on underflow (when the result is negative).
52      *
```

```
53       * Counterpart to Solidity's `-` operator.
54       *
55       * Requirements:
56       * - Subtraction cannot underflow.
57       */
58      function sub(uint256 a, uint256 b) internal pure returns (uint256) {
59          return sub(a, b, "SafeMath: subtraction underflow");
60      }
61
62      /**
63       * @dev Returns the subtraction of two unsigned integers, reverting with
    custom message on underflow (when the result is negative).
64       *
65       * Counterpart to Solidity's `-` operator.
66       *
67       * Requirements:
68       * - Subtraction cannot underflow.
69       */
70      function sub(uint256 a, uint256 b, string memory errorMessage) internal pure
    returns (uint256) {
71          require(b <= a, errorMessage);
72          uint256 c = a - b;
73
74          return c;
75      }
```

Status

ⓘ Acknowledged

[WP-N9] Unreachable code

Issue Description

L231 requires `round < completedPurchaseRounds[recipient]` so that it's impossible for `completedPurchaseRounds[recipient] == 0` .

https://github.com/kinetic-market/money-market-contracts/blob/0e5b054ae43c349750f79dd14ce005a8e06942ab/contracts/tokensale/ProtocolTokenSaleDistributor.sol#L230-L259

```
230  function _getClaimableTokenAmountPerRound(address recipient, uint round) internal
     view returns (uint) {
231      require(round < completedPurchaseRounds[recipient], "Invalid round");
232
233      if (completedPurchaseRounds[recipient] == 0) {
234          return 0;
235      }
236
237      uint initialClaimableTokens =
     initialReleasePercentages[recipient][round].mul(purchasedTokens[recipient][round]).div(100e18
238
239      uint elapsedSecondsSinceEpoch =
     block.timestamp.sub(cliffEndingEpochs[recipient][round]);
240      // Number of elapsed release periods after the initial release
241      uint elapsedVestingReleasePeriods =
     elapsedSecondsSinceEpoch.div(releasePeriodLength);
242
243      uint claimableTokensToDate = 0;
244      if (elapsedVestingReleasePeriods.add(1) >= releasePeriods[recipient][round]) {
245          claimableTokensToDate = purchasedTokens[recipient][round];
246      } else {
247          uint tokensToVest =
     purchasedTokens[recipient][round].sub(initialClaimableTokens);
248          // Multiply before division to avoid rounding errors on small numbers
249          uint claimableVestedTokens =
     tokensToVest.mul(elapsedVestingReleasePeriods).div(releasePeriods[recipient][round].sub(1));
250          claimableTokensToDate = claimableVestedTokens.add(initialClaimableTokens);
251          if (claimableTokensToDate > purchasedTokens[recipient][round]) {
252              claimableTokensToDate = purchasedTokens[recipient][round];
253          }
254      }
```

```
255
256     uint remainingClaimableTokensToDate =
    claimableTokensToDate.sub(claimedTokens[recipient][round]);
257
258     return remainingClaimableTokensToDate;
259 }
```

Status

✓ Fixed

Appendix

Timeliness of content

The content contained in the report is current as of the date appearing on the report and is subject to change without notice, unless indicated otherwise by WatchPug; however, WatchPug does not guarantee or warrant the accuracy, timeliness, or completeness of any report you access using the internet or other means, and assumes no obligation to update any information following publication.

Disclaimer

This report is based on the scope of materials and documentation provided for a limited review at the time provided. Results may not be complete nor inclusive of all vulnerabilities. The review and this report are provided on an as-is, where-is, and as-available basis. You agree that your access and/or use, including but not limited to any associated services, products, protocols, platforms, content, and materials, will be at your sole risk. Smart Contract technology remains under development and is subject to unknown risks and flaws. The review does not extend to the compiler layer, or any other areas beyond the programming language, or other programming aspects that could present security risks. A report does not indicate the endorsement of any particular project or team, nor guarantee its security. No third party should rely on the reports in any way, including for the purpose of making any decisions to buy or sell a product, service or any other asset. To the fullest extent permitted by law, we disclaim all warranties, expressed or implied, in connection with this report, its content, and the related services and products and your use thereof, including, without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement. We do not warrant, endorse, guarantee, or assume responsibility for any product or service advertised or offered by a third party through the product, any open source or third-party software, code, libraries, materials, or information linked to, called by, referenced by or accessible through the report, its content, and the related services and products, any hyperlinked websites, any websites or mobile applications appearing on any advertising, and we will not be a party to or in any way be responsible for monitoring any transaction between you and any third-party providers of products or services. As with the purchase or use of a product or service through any medium or in any environment, you should use your best judgment and exercise caution where appropriate. FOR AVOIDANCE OF DOUBT, THE REPORT, ITS CONTENT, ACCESS, AND/OR USAGE THEREOF, INCLUDING ANY ASSOCIATED SERVICES OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, INVESTMENT, TAX, LEGAL, REGULATORY, OR OTHER ADVICE.