Mainstreet v2 Audit Report

Jul 18, 2025

Table of Contents

Summary

This report has been prepared for Mainstreet 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	**Mainstreet**
Codebase	**https://github.com/Mainstreet-Labs/mainstreet-core**
Commit	**65af88a4bca55696cf5cb052363c5f91b9a6b0a5**
Language	**Solidity**

Audit Summary

Delivery Date	**Jul 18, 2025**
Audit Methodology	**Static Analysis, Manual Review**
Total Isssues	**11**

[WP-M1] When an `asset` has insufficient balance, users' pending `RedemptionRequest` will be blocked.

Medium

Issue Description

During `requestTokens()` , users' `msUSD` balance is burned.

These assets can only be redeemed through `claimTokens()` .

`claimTokens()` will be blocked and revert when the user's total `amountToClaim` is greater than `(IERC20(asset).balanceOf(address(this))` .

Users cannot cancel (no relevant external function found) a stuck `RedemptionRequest` to create new `RedemptionRequest` for other `asset` s.

Recommendation

Add cancellation functionality.

```
@@ 390,398 @@
399     function requestTokens(address asset, uint256 amount) external nonReentrant validAsset(asset, false) onlyWhitelisted {
400         if (!redemptionsEnabled) revert RedemptionsDisabled();
401
402         msUSD.burnFrom(msg.sender, amount);
403
404         uint256 amountAsset = IOracle(assetInfos[asset].oracle).amountOf(amount, maxAge, Math.Rounding.Floor);
405         amountAsset = amountAsset - (amountAsset * tax / 1000);
406         pendingClaims[asset] += amountAsset;
407
408         if (pendingClaims[asset] > redemptionCap[asset]) revert RedemptionCapExceeded(pendingClaims[asset], redemptionCap[asset]);
409
410         uint48 claimableAfter = clock() + claimDelay;
411         redemptionRequests[msg.sender].push(RedemptionRequest({
412             asset: asset,
413             amount: amountAsset,
```

```
414             claimableAfter: claimableAfter,
415             claimed: 0
416         }));
417
418         uint256 index = (redemptionRequests[msg.sender].length - 1).toUint32();
419         redemptionRequestsByAsset[msg.sender][asset].push(index);
420
421         emit TokensRequested(msg.sender, asset, index, amount, amountAsset,
    claimableAfter);
422     }
```

```
    @@ 444,451 @@
452     function claimTokens(address asset) external nonReentrant validAsset(asset,
    true) onlyWhitelisted {
453         (uint256 amountRequested, uint256 amountToClaim) = _claimTokens(asset,
    msg.sender);
454
455         if (amountToClaim == 0) revert NoTokensClaimable();
456
    amountToClaim.requireSufficientFunds(IERC20(asset).balanceOf(address(this)));
457
458         emit TokensClaimed(msg.sender, asset, amountRequested, amountToClaim);
459
460         IERC20(asset).safeTransfer(msg.sender, amountToClaim);
461         pendingClaims[asset] -= amountRequested;
462     }
```

```
    @@ 122,127 @@
128     function requireSufficientFunds(uint256 requested, uint256 available) internal
    pure {
129         if (requested > available) {
130             revert IErrors.InsufficientFunds(requested, available);
131         }
132     }
```

Status

ⓘ Acknowledged

[WP-L2] If a user accumulates too many pending `RedemptionRequest` s, the `claimTokens()` function may revert due to exceeding the gas limit, preventing the user from redeeming their msUSD back to `asset` .

Low

Issue Description

- `requestTokens()` has no limit on the number of pending `RedemptionRequest` s per user
- `claimTokens()` cannot specify the maximum number of `RedemptionRequest` s to process in one call

https://github.com/Mainstreet-Labs/mainstreet-core/tree/dev/blob/1976bc431dd35a74d71f8d85da6df12ab55eac40/src/MainstreetMinter.sol#L390-L422

```
@@ 390,398 @@
399     function requestTokens(address asset, uint256 amount) external nonReentrant validAsset(asset, false) onlyWhitelisted {
400         if (!redemptionsEnabled) revert RedemptionsDisabled();
401
402         msUSD.burnFrom(msg.sender, amount);
403
404         uint256 amountAsset = IOracle(assetInfos[asset].oracle).amountOf(amount, maxAge, Math.Rounding.Floor);
405         amountAsset = amountAsset - (amountAsset * tax / 1000);
406         pendingClaims[asset] += amountAsset;
407
408         if (pendingClaims[asset] > redemptionCap[asset]) revert RedemptionCapExceeded(pendingClaims[asset], redemptionCap[asset]);
409
410         uint48 claimableAfter = clock() + claimDelay;
411         redemptionRequests[msg.sender].push(RedemptionRequest({
412             asset: asset,
413             amount: amountAsset,
414             claimableAfter: claimableAfter,
415             claimed: 0
416         }));
417
```

```
418         uint256 index = (redemptionRequests[msg.sender].length - 1).toUint32();
419         redemptionRequestsByAsset[msg.sender][asset].push(index);
420 
421         emit TokensRequested(msg.sender, asset, index, amount, amountAsset,
    claimableAfter);
422     }
```

```
    @@ 444,451 @@
452     function claimTokens(address asset) external nonReentrant validAsset(asset,
    true) onlyWhitelisted {
453         (uint256 amountRequested, uint256 amountToClaim) = _claimTokens(asset,
    msg.sender);
454 
455         if (amountToClaim == 0) revert NoTokensClaimable();
456 
    amountToClaim.requireSufficientFunds(IERC20(asset).balanceOf(address(this)));
457 
458         emit TokensClaimed(msg.sender, asset, amountRequested, amountToClaim);
459 
460         IERC20(asset).safeTransfer(msg.sender, amountToClaim);
461         pendingClaims[asset] -= amountRequested;
462     }
463 
    @@ 464,473 @@
474     function _claimTokens(address asset, address user) internal returns (uint256
    amountRequested, uint256 amountBeingClaimed) {
475         uint256 numRequests = redemptionRequestsByAsset[user][asset].length;
476         uint256 i = firstUnclaimedIndex[user][asset];
477 
478         uint256 timestamp = clock();
479 
480         while (i < numRequests) {
481             RedemptionRequest storage userRequest =
    _unsafeRedemptionRequestByAssetAccess(
482                 redemptionRequestsByAsset[user][asset],
483                 redemptionRequests[user],
484                 i
485             );
486             if (timestamp >= userRequest.claimableAfter) {
487                 unchecked {
```

```
488                     uint256 amountClaimable = userRequest.amount *
    coverageRatio.upperLookupRecent(userRequest.claimableAfter) / 1e18;
489                     userRequest.claimed = amountClaimable;
490
491                     amountRequested += userRequest.amount;
492                     amountBeingClaimed += amountClaimable;
493
494                     firstUnclaimedIndex[user][asset] = i + 1;
495                 }
496             } else {
497                 break;
498             }
499             unchecked {
500                 ++i;
501             }
502         }
503
504         return (amountRequested, amountBeingClaimed);
505     }
```

Recommendation

Allow users to specify the number of requests to process per call.

Status

✓ Fixed

[WP-L3] `requestTokens(address asset, uint256 amount)` lacks a `minAmount` trade condition control similar to the one in `mint(address asset, uint256 amountIn, uint256 minAmountOut)`

Low

Issue Description

The `RedemptionRequest.amount` returned to users in `requestTokens(address asset, uint256 amount)` is affected by `tax` and `assetInfos[asset].oracle`, same as in `mint(address asset, uint256 amountIn, uint256 minAmountOut)`.

However, it lacks a trade condition control parameter like `uint256 minAmountOut` in `mint(address asset, uint256 amountIn, uint256 minAmountOut)`.

https://github.com/Mainstreet-Labs/mainstreet-core/tree/dev/blob/1976bc431dd35a74d71f8d85da6df12ab55eac40/src/MainstreetMinter.sol#L363-L388

```
363     /**
364      * @notice Mints msUSD tokens by accepting a deposit of approved collateral assets.
365      * @dev Executes the complete minting workflow: transfers collateral from user, applies fee deduction (if any),
366      * calculates token output via oracle price, and distributes msUSD to the msg.sender.
367      * @param asset The collateral token address used for backing the generated msUSD.
368      * @param amountIn The quantity of collateral tokens to be deposited.
369      * @param minAmountOut The minimum acceptable msUSD output, transaction reverts if not satisfied.
370      * @return amountOut The precise quantity of msUSD issued to the caller's address.
371      */
372     function mint(address asset, uint256 amountIn, uint256 minAmountOut)
373         external
374         nonReentrant
375         validAsset(asset, false)
376         onlyWhitelisted
377         returns (uint256 amountOut)
```

```
378     {
379         IERC20(asset).safeTransferFrom(msg.sender, address(this), amountIn);
380         emit CustodyTransfer(address(this), asset, amountIn);
381 
382         uint256 amountAfterTax = amountIn - (amountIn * tax / 1000);
383         amountOut = _mintTokens(asset, msg.sender, amountAfterTax);
384 
385         if (amountOut < minAmountOut) revert
    InsufficientOutputAmount(minAmountOut, amountOut);
386 
387         emit Mint(msg.sender, asset, amountIn, amountOut);
388     }
```

```
    @@ 841,850 @@
851     function _mintTokens(address asset, address recipient, uint256 amountToMint)
    internal returns (uint256 amountMinted) {
852         uint256 balanceBefore = msUSD.balanceOf(recipient);
853         msUSD.mint(recipient,
    IOracle(assetInfos[asset].oracle).valueOf(amountToMint, maxAge,
    Math.Rounding.Floor));
854         unchecked {
855             amountMinted = msUSD.balanceOf(recipient) - balanceBefore;
856         }
857     }
```

```
390     /**
391      * @notice Initiates the withdrawal process for converting msUSD back to
    underlying collateral.
392      * @dev Burns the caller's msUSD tokens and registers a time-locked claim on
    the specified asset.
393      * The system calculates equivalent collateral value using current oracle
    rates, applies
394      * the redemption fee, and schedules the claim based on configured delay
    parameters.
395      * Redemption requests are tracked both globally and per-asset for efficient
    processing.
396      * @param asset The collateral token address requested for withdrawal.
397      * @param amount The quantity of msUSD to be burned for redemption.
398      */
```

```
399     function requestTokens(address asset, uint256 amount) external nonReentrant
    validAsset(asset, false) onlyWhitelisted {
400         if (!redemptionsEnabled) revert RedemptionsDisabled();
401
402         msUSD.burnFrom(msg.sender, amount);
403
404         uint256 amountAsset = IOracle(assetInfos[asset].oracle).amountOf(amount,
    maxAge, Math.Rounding.Floor);
405         amountAsset = amountAsset - (amountAsset * tax / 1000);
406         pendingClaims[asset] += amountAsset;
407
408         if (pendingClaims[asset] > redemptionCap[asset]) revert
    RedemptionCapExceeded(pendingClaims[asset], redemptionCap[asset]);
409
410         uint48 claimableAfter = clock() + claimDelay;
411         redemptionRequests[msg.sender].push(RedemptionRequest({
412             asset: asset,
413             amount: amountAsset,
414             claimableAfter: claimableAfter,
415             claimed: 0
416         }));
417
418         uint256 index = (redemptionRequests[msg.sender].length - 1).toUint32();
419         redemptionRequestsByAsset[msg.sender][asset].push(index);
420
421         emit TokensRequested(msg.sender, asset, index, amount, amountAsset,
    claimableAfter);
422     }
```

Status

ⓘ Acknowledged

[WP-L4] `_disableInitializers()` is missing in `StakedmsUSD.constructor()`

Low

Issue Description

For comparison, `msUSDV2.sol` and `msUSDV2Satellite.sol` in the same directory have `_disableInitializers()` in their `constructor()` s.

https://github.com/Mainstreet-Labs/mainstreet-core/tree/dev/blob/1976bc431dd35a74d71f8d85da6df12ab55eac40/src/v2/StakedmsUSD.sol#L105

```
105     constructor() {}
106
        @@ 107,112 @@
113     function initialize(
        @@ 114,116 @@
117     ) public initializer {
        @@ 118,128 @@
129     }
```

Status

✓ Fixed

[WP-L5] When `available < claimable`, a `user` cannot claim tokens. In this case, `claimableTokens()` should potentially return `0` instead of `available`.

Low

Issue Description

Under such circumstances, `claimTokens()` will revert with `IErrors.InsufficientFunds(requested, available)` at L456.

https://github.com/Mainstreet-Labs/mainstreet-core/tree/dev/blob/1976bc431dd35a74d71f8d85da6df12ab55eac40/src/MainstreetMinter.sol#L424-L462

```
424     /**
425      * @notice Evaluates the maximum amount of a specific asset that can be withdrawn by a user.
426      * @dev Determines the total eligible withdrawal amount based on matured redemption requests,
427      * constrained by actual asset availability in the contract. Takes into account both
428      * time-based eligibility and current contract holdings.
429      * @param user The wallet address whose eligible withdrawals are being calculated.
430      * @param asset The collateral token address being evaluated for withdrawal.
431      * @return amount The maximum quantity currently available for withdrawal.
432      */
433     function claimableTokens(address user, address asset)
434         external
435         view
436         validAsset(asset, true)
437         returns (uint256 amount)
438     {
439         uint256 claimable = _calculateClaimableTokens(user, asset);
440         uint256 available = IERC20(asset).balanceOf(address(this));
441         return available < claimable ? available : claimable;
442     }
443
444     /**
445      * @notice Finalizes the withdrawal of previously requested collateral assets.
```

```
446       * @dev Processes all matured redemption requests for the specified asset,
447       * applies the current coverage ratio to determine final withdrawal amount,
448       * transfers the assets to the caller, and updates the global redemption
      state.
449       * Fails if no eligible tokens are available or if contract lacks sufficient
      balance.
450       * @param asset The collateral token address to be withdrawn.
451       */
452      function claimTokens(address asset) external nonReentrant validAsset(asset,
      true) onlyWhitelisted {
453          (uint256 amountRequested, uint256 amountToClaim) = _claimTokens(asset,
      msg.sender);
454
455          if (amountToClaim == 0) revert NoTokensClaimable();
456
      amountToClaim.requireSufficientFunds(IERC20(asset).balanceOf(address(this)));
457
458          emit TokensClaimed(msg.sender, asset, amountRequested, amountToClaim);
459
460          IERC20(asset).safeTransfer(msg.sender, amountToClaim);
461          pendingClaims[asset] -= amountRequested;
462      }
```

```
      @@ 122,127 @@
128      function requireSufficientFunds(uint256 requested, uint256 available) internal
      pure {
129          if (requested > available) {
130              revert IErrors.InsufficientFunds(requested, available);
131          }
132      }
```

Status

✓ Fixed

[WP-I6] Consider using LayerZero v2 based OFT.

Informational

Issue Description

This will help avoid issues like ordered delivery (blocking, where any failed message blocks the channel) in v1, and will help avoid using less maintained infrastructure.

Currently, `msUSDV2` inherits from `OFTUpgradeable`, which inherits from `OFTCoreUpgradeable`, which in turn inherits from `NonblockingLzAppUpgradeable`. These are LayerZero v1's LzApp and OFT implementations.

https://github.com/LayerZero-Labs/endpoint-v1-solidity-examples has been archived and hasn't been maintained for a long time, with a notice:

https://github.com/LayerZero-Labs/endpoint-v1-solidity-examples/blob/cdc93994911829b1348f6ac18000000a43432ef1/README.md?plain=1#L9-L16

> **LayerZero V2** is now available here, offering improvements in cross-chain transaction speed, gas efficiency, and more.
> Review the LayerZero V2 Documentation for a comprehensive overview of the new feature set.
> For these reasons, we ***recommend deploying to LayerZero V2 instead of LayerZero V1***.
> All of the contracts available in this repo should be considered legacy for Endpoint V1.

V1 has several issues, for example:

`NonblockingLzAppUpgradeable` aims to prevent blocking subsequent LayerZero message receive operations when `nonblockingLzReceive()` fails by not reverting in `_blockingLzReceive()` (which is called in `lzReceive()`), but instead stores the failed message.

However, `NonblockingLzAppUpgradeable` L92 forwards all remaining `gasleft()` to `nonblockingLzReceive()` without reserving gas for the subsequent `if (!success) { _storeFailedMessage(...); }` operation, which could cause an unexpected out of gas revert during `_storeFailedMessage(...)` execution.

Unlike LayerZero v1, LayerZero v2 defaults to Unordered Delivery:

https://docs.layerzero.network/v2/concepts/message-ordering

https://github.com/Mainstreet-Labs/mainstreet-core/tree/dev/blob/1976bc431dd35a74d71f8d85da6df12ab55eac40/src/utils/lzApp/NonblockingLzAppUpgradeable.sol#L77-L100

```
@@ 77,85 @@
86      function _blockingLzReceive(uint16 srcChainId, bytes memory srcAddress, uint64 nonce, bytes memory payload)
87          internal
88          virtual
89          override
90      {
91          (bool success, bytes memory reason) = address(this).excessivelySafeCall(
92              gasleft(),
93              150,
94              abi.encodeWithSelector(this.nonblockingLzReceive.selector, srcChainId, srcAddress, nonce, payload)
95          );
96          // try-catch all errors/exceptions
97          if (!success) {
98              _storeFailedMessage(srcChainId, srcAddress, nonce, payload, reason);
99          }
100     }
```

Recommendation

Consider using LayerZero v2 based OFT:

- https://github.com/LayerZero-Labs/LayerZero-v2/blob/main/packages/layerzero-v2/evm/oapp/contracts/oft/OFT.sol
- https://docs.layerzero.network/v2/developers/evm/oft/quickstart

Status

ⓘ Acknowledged

[WP-I7] `msUSDV2Satellite` and `msUSDV2` do not modify `OFTCoreUpgradeable.sendFrom()`, therefore overriding `sendFrom()` is unnecessary.

Informational

Issue Description

Removing it will simplify the code, especially for `msUSDV2Satellite`.

Verified with `forge build` - removal does not affect compilation.

```
31  contract msUSDV2Satellite is UUPSUpgradeable, OFTUpgradeable {
    @@ 32,57 @@
58
59      function sendFrom(
60          address _from,
61          uint16 _dstChainId,
62          bytes calldata _toAddress,
63          uint256 _amount,
64          address payable _refundAddress,
65          address _zroPaymentAddress,
66          bytes calldata _adapterParams
67      ) public payable override(IOFTCore, OFTCoreUpgradeable) {
68          _send(
69              _from,
70              _dstChainId,
71              _toAddress,
72              _amount,
73              _refundAddress,
74              _zroPaymentAddress,
75              _adapterParams
76          );
77      }
78  }
```

```
116     function sendFrom(
117         address _from,
118         uint16 _dstChainId,
119         bytes calldata _toAddress,
120         uint256 _amount,
121         address payable _refundAddress,
122         address _zroPaymentAddress,
123         bytes calldata _adapterParams
124     ) public payable override(IOFTCore, OFTCoreUpgradeable) {
125         _send(
126             _from,
127             _dstChainId,
128             _toAddress,
129             _amount,
130             _refundAddress,
131             _zroPaymentAddress,
132             _adapterParams
133         );
134     }
```

```
116     function sendFrom(
117         address from,
118         uint16 dstChainId,
119         bytes calldata toAddress,
120         uint256 amount,
121         address payable refundAddress,
122         address zroPaymentAddress,
123         bytes calldata adapterParams
124     ) public payable virtual override {
125         _send(from, dstChainId, toAddress, amount, refundAddress,
    zroPaymentAddress, adapterParams);
126     }
```

Status

ⓘ Acknowledged

[WP-N8] Unused private contract scoped variables `_scaleUp` and `_scalePrecision`

Issue Description

The private contract scoped variables `_scaleUp` and `_scalePrecision` are only used in `StaticPriceOracle.constructor()`. These variables are not used anywhere else in the `StaticPriceOracle` contract.

```
@@ 9,16 @@
17  contract StaticPriceOracle is IOracle {
18      using Math for uint256;
19
20      address public immutable token;
21
22      bool private immutable _scaleUp;
23      uint256 private immutable _scalePrecision;
24      uint256 private immutable _tokenPrecision;
25      uint256 private immutable _staticPrice;
26
@@ 27,33 @@
34      constructor(address _token, uint256 staticPrice, uint8 decimals) {
35          token = _token;
36          _tokenPrecision = 10 ** IERC20Metadata(_token).decimals();
37          uint256 pricePrecision = 10 ** decimals;
38          require(pricePrecision <= 1e18, "OracleWrapper: too many decimals");
39          bool scaleUp = pricePrecision < 1e18;
40          _scaleUp = scaleUp;
41          _scalePrecision = scaleUp ? 1e18 / pricePrecision : 1;
42          _staticPrice = staticPrice * _scalePrecision;
43      }
44
@@ 45,175 @@
176 }
```

Recommendation

Consider changing to something like:

```
17  contract StaticPriceOracle is IOracle {
18      using Math for uint256;
19
20      address public immutable token;
21      uint256 private immutable _tokenPrecision;
22      uint256 private immutable _staticPrice;
23

    @@ 24,30 @@

31      constructor(address _token, uint256 staticPrice, uint8 decimals) {
32          token = _token;
33          _tokenPrecision = 10 ** IERC20Metadata(_token).decimals();
34          require(decimals <= 18, "OracleWrapper: too many decimals");
35          _staticPrice = decimals < 18 ? staticPrice * 10 ** (18 - decimals) :
    staticPrice;
36      }
37

    @@ 38,168 @@

169 }
```

Status

ⓘ Acknowledged

[WP-N9] Unused library SafeERC20

Issue Description

No functions from `SafeERC20` are used in `msUSDSilo` .

https://github.com/Mainstreet-Labs/mainstreet-core/tree/dev/blob/1976bc431dd35a74d71f8d85da6df12ab55eac40/src/v2/msUSDSilo.sol

```
1  // SPDX-License-Identifier: GPL-3.0
2  pragma solidity ^0.8.0;
3
4  import "@openzeppelin/contracts/token/ERC20/IERC20.sol";
5  import "@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol";
6
   @@ 7,10 @@
11 contract msUSDSilo {
12     using SafeERC20 for IERC20;
13
   @@ 14,32 @@
33 }
```

Status

ⓘ Acknowledged

[WP-O10] When cooldown is on, only the `owner` of the shares can `unstake` .

Issue Description

If cooldown is off, any authorized `caller` with sufficient allowance can call `withdraw` or `redeem` without needing the `owner` 's permission.

This may or may not be expected.

```
217  /**
218       * @notice Allows users to claim their assets after the cooldown period has
     ended
219       * @dev Can be called by anyone to claim their own assets. The cooldown must
     have expired
220       * and the user must have assets in cooldown. Transfers assets from silo to
     receiver.
221       * @param receiver Address to send the assets to
222       */
223      function unstake(address receiver) external {
224          UserCooldown storage userCooldown = cooldowns[msg.sender];
225          uint256 assets = userCooldown.underlyingAmount;
226
227          if (userCooldown.cooldownEnd > block.timestamp) revert
     CooldownNotFinished(block.timestamp, userCooldown.cooldownEnd);
228          if (assets == 0) revert NothingToUnstake();
229
230          emit Unstake(msg.sender, receiver, assets);
231
232          userCooldown.cooldownEnd = 0;
233          userCooldown.underlyingAmount = 0;
234
235          silo.withdraw(receiver, assets);
236      }
237
238      /**
239       * @notice redeem assets and starts a cooldown to claim the converted
     underlying asset
240       * @param assets assets to redeem
241       * @param owner address to redeem and start cooldown, owner must allowed
     caller to perform this action
```

```
242         */
243         function cooldownAssets(
244             uint256 assets,
245             address owner
246         ) external ensureCooldownOn returns (uint256) {
247             if (assets > maxWithdraw(owner)) revert ExcessiveWithdrawAmount();
248 
249             uint256 shares = previewWithdraw(assets);
250 
251             cooldowns[owner].cooldownEnd = uint104(block.timestamp) +
    cooldownDuration;
252             cooldowns[owner].underlyingAmount += assets;
253 
254             _withdraw(_msgSender(), address(silo), owner, assets, shares);
255 
256             return shares;
257         }
```

Status

ⓘ Acknowledged

[WP-G11] `MainstreetMinter.requestTokens()` redundant safe cast to uint32

Gas

Issue Description

On L418, `uint256` is safely casted to `uint32` and then assigned back to `uint256` type variables `index` and `redemptionRequestsByAsset[msg.sender][asset]` .

https://github.com/Mainstreet-Labs/mainstreet-core/tree/dev/blob/1976bc431dd35a74d71f8d85da6df12ab55eac40/src/MainstreetMinter.sol#L390-L422

```
390     /**
391      * @notice Initiates the withdrawal process for converting msUSD back to underlying collateral.
392      * @dev Burns the caller's msUSD tokens and registers a time-locked claim on the specified asset.
393      * The system calculates equivalent collateral value using current oracle rates, applies
394      * the redemption fee, and schedules the claim based on configured delay parameters.
395      * Redemption requests are tracked both globally and per-asset for efficient processing.
396      * @param asset The collateral token address requested for withdrawal.
397      * @param amount The quantity of msUSD to be burned for redemption.
398      */
399     function requestTokens(address asset, uint256 amount) external nonReentrant validAsset(asset, false) onlyWhitelisted {
400         if (!redemptionsEnabled) revert RedemptionsDisabled();
401 
402         msUSD.burnFrom(msg.sender, amount);
403 
404         uint256 amountAsset = IOracle(assetInfos[asset].oracle).amountOf(amount, maxAge, Math.Rounding.Floor);
405         amountAsset = amountAsset - (amountAsset * tax / 1000);
406         pendingClaims[asset] += amountAsset;
407 
408         if (pendingClaims[asset] > redemptionCap[asset]) revert RedemptionCapExceeded(pendingClaims[asset], redemptionCap[asset]);
409 
```

```
410         uint48 claimableAfter = clock() + claimDelay;
411         redemptionRequests[msg.sender].push(RedemptionRequest({
412             asset: asset,
413             amount: amountAsset,
414             claimableAfter: claimableAfter,
415             claimed: 0
416         }));
417
418         uint256 index = (redemptionRequests[msg.sender].length - 1).toUint32();
419         redemptionRequestsByAsset[msg.sender][asset].push(index);
420
421         emit TokensRequested(msg.sender, asset, index, amount, amountAsset,
    claimableAfter);
422     }
```

```
56      /// @dev Stores the indexes for each redemption request accoridng to user and
    asset.
57      mapping(address user => mapping(address asset => uint256[])) public
    redemptionRequestsByAsset;
```

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.