Findings for Notional x Index Coop C4 Contest

Type of Audit: Security Review
Type of Project: DeFi
Language: Solidity
Methods: Manual review, automated test suite

Audit Scope

1. `NotionalTradeModule`
2. `wfCashLogic`

---

1. Contract `NotionalTradeModule.sol`

[LOW] Use of non-existent function `calculateAndEditDefaultPosition` in `_updateSetTokenPositions` function

File: L664 L670
Description: `calculateAndEditDefaultPosition` function does not exist in `ISetToken.sol` interface. However, it is called in `_updateSetTokenPositions` function. This might flaw the logic of other functions relying on this `_updateSetTokenPositions` function.

```
function _updateSetTokenPositions(
    ISetToken setToken,
    address sendToken,
    uint256 preTradeSendTokenBalance,
    address receiveToken,
    uint256 preTradeReceiveTokenBalance
) internal returns (uint256, uint256) {

    uint256 setTotalSupply = setToken.totalSupply();

    (uint256 currentSendTokenBalance,,) = setToken.calculateAndEditDefaultPosi
        sendToken,
        setTotalSupply,
        preTradeSendTokenBalance
    );

    (uint256 currentReceiveTokenBalance,,) = setToken.calculateAndEditDefaultP
        receiveToken,
        setTotalSupply,
        preTradeReceiveTokenBalance
    );

    return (
        preTradeSendTokenBalance.sub(currentSendTokenBalance),
        currentReceiveTokenBalance.sub(preTradeReceiveTokenBalance)
    );
}
```

Recommendation: Consider adding `calculateAndEditDefaultPosition` as an external function in `ISetToken.sol` interface.

[LOW] Missing event emission for `_updateSetTokenPositions` function

File: L654
Description: `_updateSetTokenPositions` do not emit appropriate event as shown below:

```
function _updateSetTokenPositions(
    ISetToken setToken,
    address sendToken,
    uint256 preTradeSendTokenBalance,
    address receiveToken,
    uint256 preTradeReceiveTokenBalance
) internal returns (uint256, uint256) {

    uint256 setTotalSupply = setToken.totalSupply();

    (uint256 currentSendTokenBalance,,) = setToken.calculateAndEditDefaultPosi
        sendToken,
        setTotalSupply,
        preTradeSendTokenBalance
    );

    (uint256 currentReceiveTokenBalance,,) = setToken.calculateAndEditDefaultP
        receiveToken,
        setTotalSupply,
        preTradeReceiveTokenBalance
    );

    return (
        preTradeSendTokenBalance.sub(currentSendTokenBalance),
        currentReceiveTokenBalance.sub(preTradeReceiveTokenBalance)
    );
}
```

Recommendation: Consider creating and emitting appropriate event for `_updateSetTokenPositions` function

[LOW] Use of Different Compiler Versions

File: L19
Description: Whereas the imported contracts use a higher compiler version, `NotionalTradeModule.sol` use a far lesser compiler version which might introduce bugs that affect the contract system negatively

```
// SPDX-License-Identifier: MIT
pragma solidity 0.6.10;
```

Recommendation: Consider using consistent compiler version

[LOW] Missing event emission for `setRedeemToUnderlying` function

File: L294
Description: State-changing function `setRedeemToUnderlying` do not emit appropriate event as shown below:

```
function setRedeemToUnderlying(
        ISetToken _setToken,
        bool _toUnderlying
    )
    external
    onlyManagerAndValidSet(_setToken)
    {
        redeemToUnderlying[_setToken] = _toUnderlying;
    }
```

Recommendation: Consider creating and emitting appropriate event for `setRedeemToUnderlying` function

[LOW] Use of OpenZeppelin's `SafeMath` arithmetic operations' `sub` function in `_updateSetTokenPositions` function without direct library import in the `NotionalTradeModule.sol` contract

File: L677 L678
Description: Openzeppelin's `SafeMath` library's `sub` function is being used in `_updateSetTokenPositions` function without `SafeMath` library being directly imported and declared to be used in `NotionalTradeModule.sol` contract thus making the contract vulnerable to overflow risks.

```
function _updateSetTokenPositions(
    ISetToken setToken,
    address sendToken,
    uint256 preTradeSendTokenBalance,
    address receiveToken,
    uint256 preTradeReceiveTokenBalance
) internal returns (uint256, uint256) {

    uint256 setTotalSupply = setToken.totalSupply();

    (uint256 currentSendTokenBalance,,) = setToken.calculateAndEditDefaultPosi
        sendToken,
        setTotalSupply,
        preTradeSendTokenBalance
    );

    (uint256 currentReceiveTokenBalance,,) = setToken.calculateAndEditDefaultP
        receiveToken,
        setTotalSupply,
        preTradeReceiveTokenBalance
    );

    return (
        preTradeSendTokenBalance.sub(currentSendTokenBalance),
        currentReceiveTokenBalance.sub(preTradeReceiveTokenBalance)
    );
}
```

Recommendation: Consider importing Openzeppelin's `SafeMath` library in `NotionalTradeModule.sol` contract and using it thus `using SafeMath for uint256;` so that `setRedeemToUnderlying` function can utilize the library's `sub` function in order to mitigate overflow risks.

[LOW] No input validation in `mintFCashPosition(...)`, `redeemFCashPosition(...)`

File(s): `index-coop-notional-trade-module/contracts/protocol/modules/v1/NotionalTradeModule.sol`

Description: Missing input validation check for `mintFCashPosition` function's `_currencyId` and `_maturity` parameters as well as `redeemFCashPosition` function's `_receiveToken` and `_sendToken` parameters. The code is shown below:

```
function mintFCashPosition(
        ISetToken _setToken,
        uint16 _currencyId,
        uint40 _maturity,
        uint256 _mintAmount,
        address _sendToken,
        uint256 _maxSendAmount
    )
        external
        nonReentrant
        onlyManagerAndValidSet(_setToken)
        returns(uint256)
    {
        require(_setToken.isComponent(address(_sendToken)), "Send token must b

        IWrappedfCashComplete wrappedfCash = _deployWrappedfCash(_currencyId,
        return _mintFCashPosition(_setToken, wrappedfCash, IERC20(_sendToken),
    }


    function redeemFCashPosition(
        ISetToken _setToken,
        uint16 _currencyId,
        uint40 _maturity,
        uint256 _redeemAmount,
        address _receiveToken,
        uint256 _minReceiveAmount
    )
        external
        nonReentrant
        onlyManagerAndValidSet(_setToken)
        returns(uint256)
    {
        IWrappedfCashComplete wrappedfCash = _getWrappedfCash(_currencyId, _ma
        require(_setToken.isComponent(address(wrappedfCash)), "fCash to redeem

        return _redeemFCashPosition(_setToken, wrappedfCash, IERC20(_receiveTo
    }
```

Recommendation: Consider adding appropriate validation logic for `_currencyid` and `_maturity` parameters of `mintFCashPosition` as well as `_receiveToken` and `_sendToken` parameters of `redeemFCashPosition` function.

2. Contract `wfCashLogic.sol`

[Low] Missing input validation in `mintViaUnderlying(...)`

File(s): (https://github.com/code-423n4/2022-06-notional-coop/blob/6f8c325f604e2576ef2575d6b57892ca181509a/notional-wrapped-fcash/contracts/wfCashLogic.sol#L27)

Description: The function `mintViaUnderlying(...)` does not validate the input parameter `depositAmountExternal`.

```
function mintViaUnderlying(
        uint256 depositAmountExternal,
        uint88 fCashAmount,
        address receiver,
        uint32 minImpliedRate
    ) external override {
        _mintInternal(depositAmountExternal, fCashAmount, receiver, minImplied
    }
```

Recommendation: Consider adding validation logic for `depositAmountExternal` parameter of `mintViaUnderlying` function

---

3. Contract `wfCashERC4626.sol`

[Low] Missing input validation in `redeem(...)`

File(s): (https://github.com/code-423n4/2022-06-notional-coop/blob/6f8c325f604e2576ef2575d6b57892ca181509a/notional-wrapped-fcash/contracts/wfCashERC4626.sol#L205)

Description: The function `redeem(...)` lacks validation logic for the `receiver` input parameter against address(0).

```
function redeem(
        uint256 shares,
        address receiver,
        address owner
    ) public override returns (uint256) {
        // It is more accurate and gas efficient to check the balance of the
        // receiver here than rely on the previewRedeem method.
        uint256 balanceBefore = IERC20(asset()).balanceOf(receiver);

        if (msg.sender != owner) {
            _spendAllowance(owner, msg.sender, shares);
        }
        _redeemInternal(shares, receiver, owner);

        uint256 balanceAfter = IERC20(asset()).balanceOf(receiver);
        uint256 assets = balanceAfter - balanceBefore;
        emit Withdraw(msg.sender, receiver, owner, assets, shares);
        return assets;
    }
```

Recommendation: Consider validating the input `receiver` parameter.

[LOW] Inconsistent import of `ReentrancyGuard` library

File(s): (https://github.com/code-423n4/2022-06-notional-coop/blob/6f8c325f604e2576ef2575d6b57892ca181509a/notional-wrapped-fcash/contracts/wfCashLogic.sol#L6)

Description: ReentrancyGaurd is not available in the path that the current import is assuming for the project.

```
import "@openzeppelin/contracts/security/ReentrancyGuard.sol";
```

Recommendation: Consider using consistent Openzeppelin version while importing contracts across all the files to mitigate such inconsistent imports

---