AUDIT REPORT V1.0

for

Fortis

by

Bichistriver

Fri, 3rd Jan, 2025

Disclaimer:

This audit report is intended for informational purposes only and does not constitute financial, legal, or investment advice. The findings and recommendations presented herein are based on a thorough analysis of the provided contracts at the time of the audit but cannot guarantee the absence of vulnerabilities, errors, or other unforeseen issues in the code.

This audit covers only the specific versions of the contracts reviewed (`FUSD.sol`, `Router.sol`, `Fortis.sol`, `Manager.sol`, and related components) as of the date of this report. Changes made to the codebase after this review are not included in the findings.

The audit does not assess the project team's operational practices, tokenomics, or compliance with regulatory requirements.

Diagram

```
                            +-------------------+
                            |       Users       |
                            +---------+---------+
                                      |
                                      |
                             +--------v--------+
                             |     Router      |
                             +--------+--------+
                                      |
            +-------------------------+---------------------------+
            |                         |                           |
            |                         |                           |
+-----------v-----------+   +---------v---------+     +-----------v-----------+
|      Manager.sol      |   |       FUSD.sol    |     |       Fortis.sol      |
|                       |   | (Fortis USD Token)|     | (Fortis ERC20 Token)  |
| - Deposit/Withdraw    |   | - Mint/Burn FUSD  |     +-----------------------+
| - Mint/Burn FUSD      |   +-------------------+
| - Handle Liquidations |
| - Manage Collateral   |
+-----------+-----------+
            ^
            |
            |
    +----------------+
    |    Oracle      |
    |                |
    | - Asset Prices |
    | - Conversion   |
    |   Rates        |
    +----------------+
```

contracts/src/FUSD.sol

[H1] - No Transfer of Ownership, which means if anything happens in the future, the mint and burn are in the hands of uncertainty

Both `mint` and `burn` functions are restricted to the contract owner (`onlyOwner` modifier). The contract lacks mechanisms to transfer or renounce ownership, which means the control of minting and burning is fixed and cannot be altered in the event of the owner's long-term unavailability or compromise.

The Fortis USD contract, in its current form, presents risks due to its centralized control and lack of upgradability and ownership transfer mechanisms.

Mitigation

Implement Ownership transfer function.

General comments

The contract is simple.

contracts/src/Router.sol

[H2] - Lack of Input Validation

Initially, the `depositAndMint` function lacked checks for non-zero input values for `assets` and `amount`, as well as validation for non-zero addresses for `receiver` and `params.owner`.

Without these checks, the function could execute meaningless transactions (e.g., depositing 0 assets), or interact with zero addresses, leading to loss of funds or tokens.

Mitigation

Import Errors Library and add these to `depositAndMint`

```
        require(receiver != address(0), Errors.ZERO_ADDRESS);
        require(params.owner != address(0), Errors.ZERO_ADDRESS);
        require(assets > 0, Errors.ZERO_ASSETS);
        require(amount > 0, Errors.ZERO_ASSETS);
```

General comments

Zero comments in the code!

contracts/libraries/ErrorsLib.sol

[QA1] - Typographical Error:

The message `INVALID_SIGNATRURE` contains a typographical error in the word "signature".

Recommendation:

Correct the spelling to `"INVALID_SIGNATURE"` to maintain professionalism and clarity in error messaging.

contracts/src/Manager.sol

[M1] there is no validation on the new `_feeReceiver` address, enabling it to be set to `address(0)`

The `Manager.sol` contract contains a function `setFeeReceiver(address _feeReceiver)` that allows the owner to modify the recipient of the protocol's fees. Currently, there is **no** validation on the new `_feeReceiver` address, enabling it to be set to `address(0)`. If fees or yield shares are minted or transferred to this zero address, those funds are effectively burned and become unrecoverable. This undermines the protocol's fee model, potentially preventing the protocol from collecting revenue for ongoing operations or compensating necessary stakeholders.

Mitigation

Add a check,

```
function setFeeReceiver(address _feeReceiver) external onlyOwner {
    require(_feeReceiver != address(0), Errors.ZERO_ADDRESS);
    feeReceiver = _feeReceiver;
}
```

[M2] No Slippage Protection in `withdraw`/`redeem` Functions of the Manager.sol

When a user who has deposited assets into the the Vault wishes to withdraw (or redeem) them, they can do so by calling either the `withdraw` or `redeem` functions.

Under normal conditions in the vault, users expect to receive their full deposited asset amount back, as there is a 1:1 exchange ratio between the asset amount and shares.

However, if the underlying yield vault experiences a loss, this exchange rate will decrease.

Mitigation

Parameters for Slippage Control:

- For the `withdraw` function, introduce a `maxSharesIn` parameter.
- For the `redeem` function, introduce a `minAssetsOut` parameter.

Validation:

- Check that the `shares` burned in `withdraw` do not exceed `maxSharesIn`.
- Ensure the `assets` received in `redeem` meet or exceed `minAssetsOut`.

Modify Functions: Update the existing `withdraw` and `redeem` functions to include these parameters and add checks.

Withdraw:

```
function withdraw(
    uint assets,
    address receiver,
    address owner,
    uint maxSharesIn // New parameter for slippage control
) {
...
}
```

Redeem:

```
function redeem(
    uint shares,
    address receiver,
    address owner,
    uint minAssetsOut // New parameter for slippage control
) {
...
}
```

Error Handling:

- Use descriptive error messages in `ErrorsLib.sol` (e.g., `EXCESSIVE_SHARES_BURNED` and `INSUFFICIENT_ASSETS_RETURNED`).