Smart Contract Security Audit Report

Table Of Contents

1 Executive Summary

On 2025.12.01, the SlowMist security team received the BitFi team's security audit application for BitFi USD, developed the audit plan according to the agreement of both parties and the characteristics of the project, and finally issued the security audit report.

The SlowMist security team adopts the strategy of "white box lead, black, grey box assists" to conduct a complete security test on the project in the way closest to the real attack.

The test method information:

Test method	Description
Black box testing	Conduct security tests from an attacker's perspective externally.
Grey box testing	Conduct security testing on code modules through the scripting tool, observing the internal running status, mining weaknesses.
White box testing	Based on the open source code, non-open source code, to detect whether there are vulnerabilities in programs such as nodes, SDK, etc.

The vulnerability severity level information:

Level	Description
Critical	Critical severity vulnerabilities will have a significant impact on the security of the DeFi project, and it is strongly recommended to fix the critical vulnerabilities.
High	High severity vulnerabilities will affect the normal operation of the DeFi project. It is strongly recommended to fix high-risk vulnerabilities.
Medium	Medium severity vulnerability will affect the operation of the DeFi project. It is recommended to fix medium-risk vulnerabilities.
Low	Low severity vulnerabilities may affect the operation of the DeFi project in certain scenarios. It is suggested that the project team should evaluate and consider whether these vulnerabilities need to be fixed.
Weakness	There are safety risks theoretically, but it is extremely difficult to reproduce in engineering.
Suggestion	There are better practices for coding or architecture.

2 Audit Methodology

The security audit process of SlowMist security team for smart contract includes two steps:

- Smart contract codes are scanned/tested for commonly known and more specific vulnerabilities using automated analysis tools.
- Manual audit of the codes for security issues. The contracts are manually analyzed to look for any potential problems.

Following is the list of commonly known vulnerabilities that was considered during the audit of the smart contract:

Serial Number	Audit Class	Audit Subclass
1	Overflow Audit	-
2	Reentrancy Attack Audit	-
3	Replay Attack Audit	-
4	Flashloan Attack Audit	-
5	Race Conditions Audit	Reordering Attack Audit
6	Permission Vulnerability Audit	Access Control Audit
		Excessive Authority Audit
7	Security Design Audit	External Module Safe Use Audit
		Compiler Version Security Audit
		Hard-coded Address Security Audit
		Fallback Function Safe Use Audit
		Show Coding Security Audit
		Function Return Value Security Audit
		External Call Function Security Audit

Serial Number	Audit Class	Audit Subclass
7	Security Design Audit	Block data Dependence Security Audit
		tx.origin Authentication Security Audit
8	Denial of Service Audit	-
9	Gas Optimization Audit	-
10	Design Logic Audit	-
11	Variable Coverage Vulnerability Audit	-
12	"False Top-up" Vulnerability Audit	-
13	Scoping and Declarations Audit	-
14	Malicious Event Log Audit	-
15	Arithmetic Accuracy Deviation Audit	-
16	Uninitialized Storage Pointer Audit	-

3 Project Overview

3.1 Project Introduction

This is a stablecoin protocol that mainly includes the Token, Staking, Manager, Minter, and Zap modules.

3.2 Vulnerability Information

The following is the status of the vulnerabilities found in this audit:

NO	Title	Category	Level	Status
N1	Cross-chain minting bypasses supply cap	Design Logic Audit	Information	Acknowledged
N2	Fund loss risk in redeem via	Design Logic Audit	Medium	Acknowledged

NO	Title	Category	Level	Status
	uninitialized epochs			
N3	Risk of excessive privilege	Authority Control Vulnerability Audit	Medium	Acknowledged
N4	Missing zero address check	Others	Suggestion	Acknowledged
N5	Missing event log	Others	Suggestion	Acknowledged
N6	Ignore function return value	Others	Suggestion	Acknowledged
N7	Inaccurate error type	Design Logic Audit	Suggestion	Acknowledged
N8	Redundant code	Others	Suggestion	Acknowledged

4 Code Overview

4.1 Contracts Description

https://github.com/bitfi-labs/contract/tree/master/contracts/bfusd

Initial audit version: 17a8354ddef7b2e69a965beacb3bc3beafccf11d is converted to

378ccff38edb462a3957b229fea56e597196914d

Final audit version: 378ccff38edb462a3957b229fea56e597196914d

Audit Scope:

```
./contracts/bfusd
├── BfusdMerkleStash.sol
├── BitFiStablecoin.sol
├── BitFiStablecoinManager.sol
├── BitFiStablecoinMinter.sol
├── BitFiStablecoinZap.sol
├── libraries
│   ├── FeeLibrary.sol
│   └── PriceLibrary.sol
├── redeemers
│   ├── BitFiStablecoinInstantRedeemer.sol
│   └── BitFiStablecoinStandardRedeemer.sol
```

```
├── staking
│   └── StakedBitFiStablecoin.sol
└── utils
    └── BitFiPausable.sol
```

The main network address of the contract is as follows:

The code was not deployed to the mainnet.

4.2 Visibility Description

The SlowMist Security team analyzed the visibility of major contracts during the audit, the result as follows:

BfusdMerkleStash			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
initialize	External	Can Modify State	initializer
_authorizeUpgrade	Internal	Can Modify State	onlyOwner
addStash	External	Can Modify State	-
setStakingPool	External	Can Modify State	onlyOwner
claim	External	Can Modify State	whenNotPaused

BitFiStablecoin			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	OFTUpgradeable
initialize	External	Can Modify State	initializer
_authorizeUpgrade	Internal	Can Modify State	onlyOwner
setMinter	External	Can Modify State	onlyOwner
increaseMinterLimit	External	Can Modify State	onlyOwner
setRedeemer	External	Can Modify State	onlyOwner

BitFiStablecoin			
setManager	External	Can Modify State	onlyOwner
setCrossChainFee	External	Can Modify State	onlyOwner
_calculateDebitAmounts	Internal	-	-
mint	External	Can Modify State	-
burn	External	Can Modify State	onlyRedeemer
decimals	Public	-	-
_update	Internal	Can Modify State	whenNotPaused
_debitView	Internal	-	-
_debit	Internal	Can Modify State	-

BitFiStablecoinManager			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
initialize	External	Can Modify State	initializer
_authorizeUpgrade	Internal	Can Modify State	onlyOwner
_checkMultisig	Internal	-	-
claimMultiWithdrawals	External	Can Modify State	-
advanceEpoch	External	Can Modify State	-
addStash	External	Can Modify State	-
_updatePoolRatio	Internal	Can Modify State	-
setMultisig	External	Can Modify State	onlyOwner
setMinEpochPeriod	External	Can Modify State	onlyOwner
setFeeReceiver	External	Can Modify State	onlyOwner

BitFiStablecoinManager			
setFeeWhitelist	External	Can Modify State	onlyOwner
setStakingPool	External	Can Modify State	onlyOwner
setBlacklisted	External	Can Modify State	onlyOwner
setMerkleStash	External	Can Modify State	onlyOwner

BitFiStablecoinMinter			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
initialize	External	Can Modify State	initializer
_authorizeUpgrade	Internal	Can Modify State	onlyOwner
mint	External	Can Modify State	whenNotPaused
previewMint	External	-	-
_calculateMint	Internal	-	-
setFees	External	Can Modify State	onlyOwner
setStalePriceDelay	External	Can Modify State	onlyOwner
setMinMintAmount	External	Can Modify State	onlyOwner
setCustodyWallet	External	Can Modify State	onlyOwner
setUnderlyingCap	External	Can Modify State	onlyOwner

BitFiStablecoinZap			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
initialize	External	Can Modify State	initializer

BitFiStablecoinZap			
_authorizeUpgrade	Internal	Can Modify State	onlyOwner
previewStake	External	-	-
stake	External	Can Modify State	-

BitFiStablecoinInstantRedeemer			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
initialize	External	Can Modify State	initializer
_authorizeUpgrade	Internal	Can Modify State	onlyOwner
redeem	External	Can Modify State	whenNotPaused
previewRedeem	External	-	-
_calculateRedemption	Internal	-	-
setFees	External	Can Modify State	onlyOwner
setStalePriceDelay	External	Can Modify State	onlyOwner
setMinRedeemAmount	External	Can Modify State	onlyOwner
setCustody	External	Can Modify State	onlyOwner
settlementDelay	External	-	-

BitFiStablecoinStandardRedeemer			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
initialize	External	Can Modify State	initializer
_authorizeUpgrade	Internal	Can Modify State	onlyOwner

BitFiStablecoinStandardRedeemer			
requestRedemption	External	Can Modify State	whenNotPaused
claimRedemption	External	Can Modify State	whenNotPaused
previewRedeem	External	-	-
_calculateRedemption	Internal	-	-
setFees	External	Can Modify State	onlyOwner
setSettlementDelay	External	Can Modify State	onlyOwner
setMinRedeemAmount	External	Can Modify State	onlyOwner
setCustody	External	Can Modify State	onlyOwner

StakedBitFiStablecoin			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	OFTUpgradeable
initialize	External	Can Modify State	initializer
initializeV2	External	Can Modify State	reinitializer(2)
_authorizeUpgrade	Internal	Can Modify State	onlyOwner
deposit	External	Can Modify State	-
mint	Public	Can Modify State	-
_deposit	Internal	Can Modify State	-
withdraw	Public	Can Modify State	-
redeem	Public	Can Modify State	-
_withdraw	Internal	Can Modify State	-
_convertToSharesDeposit	Internal	-	-
_convertToAssetsDeposit	Internal	-	-

StakedBitFiStablecoin			
_convertToSharesCurrent	Internal	-	-
_convertToAssets	Internal	-	-
convertToShares	Public	-	-
convertToAssets	Public	-	-
asset	Public	-	-
totalAssets	Public	-	-
maxDeposit	Public	-	-
maxMint	Public	-	-
maxWithdraw	Public	-	-
maxRedeem	Public	-	-
previewDeposit	Public	-	-
previewMint	Public	-	-
previewWithdraw	Public	-	-
previewRedeem	Public	-	-
claimWithdrawals	External	Can Modify State	-
claimWithdrawalsFor	External	Can Modify State	-
_claimWithdrawals	Internal	Can Modify State	-
_currentRatio	Internal	-	-
currentRatio	Public	-	-
setRatio	External	Can Modify State	-
_update	Internal	Can Modify State	whenNotPaused
setFees	External	Can Modify State	onlyOwner

StakedBitFiStablecoin			
setCrossChainFee	External	Can Modify State	onlyOwner
setSettlementDelay	External	Can Modify State	onlyOwner
setDepositCap	External	Can Modify State	onlyOwner
_calculateCrossChainDebitAmounts	Internal	-	-
_debitView	Internal	-	-
_debit	Internal	Can Modify State	-
_credit	Internal	Can Modify State	-
decimals	Public	-	-

BitFiPausable			
Function Name	Visibility	Mutability	Modifiers
pause	Public	Can Modify State	onlyOwner
unpause	Public	Can Modify State	onlyOwner

4.3 Vulnerability Summary

[N1] [Information] Cross-chain minting bypasses supply cap

Category: Design Logic Audit

Content

In the StakedBitFiStablecoin contract, the `_credit` function mints new StakedBitFiStablecoin tokens when processing cross-chain transfers but fails to verify if the operation causes the `totalSupply()` to exceed the configured `supplyCap`. Unlike local `deposit` or `mint` functions which strictly enforce this limit, the cross-chain logic allows the supply cap to be circumvented, potentially breaking protocol invariants and risk management controls.

- contracts/bfusd/staking/StakedBitFiStablecoin.sol#L418-L434

```
    function _credit(
        address _to,
        uint256 _amountLD,
        uint32 // srcEid
    ) internal virtual override returns (uint256 amountReceivedLD) {
        amountReceivedLD = _amountLD;

        // Record the mint amount in the manager for settlement at epoch end
        if (_amountLD > 0) {
            manager.recordCrossChainMint(_amountLD);
        }

        // Mint the staked tokens
        _mint(_to, _amountLD);``````

        return amountReceivedLD;
    }
```

Solution

Add a check in the _credit function to verify that the new total supply does not exceed the configured supply cap before minting tokens.

Status

Acknowledged

[N2] [Medium] Fund loss risk in redeem via uninitialized epochs

Category: Design Logic Audit

Content

If the `advanceEpoch` function in the BitFiStablecoinManager contract fails to update an active staking pool, the pool's ratio for the new epoch will remain zero. In this state, the `deposit`, `mint`, and `withdraw` functions in the StakedBitFiStablecoin contract will revert due to internal zero-value checks. However, the `redeem` function lacks a check for `assets > 0`, allowing the transaction to succeed: the user's shares are burned, but they receive a withdrawal request with an amount of 0, resulting in the permanent loss of their principal.

- contracts/bfusd/BitFiStablecoinManager.sol#L101-L116

```
    function advanceEpoch(IStakedBitFiStablecoin[] calldata _pools, uint256[]
calldata _newRatios) external {
        _checkMultisig();
```

```
        if (block.timestamp < lastEpochAdvanceTime + minEpochPeriod) revert
InvalidData();
        if (_pools.length != _newRatios.length) revert InvalidData();
        uint256 epoch = ++currentEpoch;
        lastEpochAdvanceTime = block.timestamp;
        uint256 endTime = block.timestamp + minEpochPeriod;

        for (uint256 i; i < _pools.length; ++i) {
            IStakedBitFiStablecoin _pool = _pools[i];
            uint256 _newRatio = _newRatios[i];
            _updatePoolRatio(_pool, epoch, _newRatio, endTime);
        }

        emit EpochAdvanced(epoch);
    }
```

Solution

Modify advanceEpoch to iterate through and update all registered active staking pools automatically, removing the dependency on external input for pool selection.

Status

Acknowledged

[N3] [Medium] Risk of excessive privilege

Category: Authority Control Vulnerability Audit

Content

1.In the BfusdMerkleStash contract, the `Owner` role has the authority to modify important contract variables, pause/unpause the contract, and upgrade the contract through the UUPS upgrade mechanism.

- contracts/bfusd/BfusdMerkleStash.sol#L65, L76-L78

```
    function _authorizeUpgrade(address newImplementation) internal override onlyOwner
{}

    function setStakingPool(address _pool) external onlyOwner {}
```

2.In the BitFiStablecoinZap contract, the `Owner` role has the authority to upgrade the contract through the UUPS upgrade mechanism.

- contracts/bfusd/BitFiStablecoinZap.sol#L42

```
    function _authorizeUpgrade(address newImplementation) internal override onlyOwner
{}
```

3.In the BitFiStablecoinMinter contract, the `Owner` role has the authority to upgrade the contract through the UUPS upgrade mechanism and set important contract variables.

- contracts/bfusd/BitFiStablecoinMinter.sol#L76, L131-L133, L135-L137, L139-L141, L143-L146, L148-L152

```
    function _authorizeUpgrade(address newImplementation) internal override onlyOwner
{}

    function setFees(FeeLibrary.Fees calldata _fees) external onlyOwner {}

    function setStalePriceDelay(uint256 _delay) external onlyOwner {}

    function setMinMintAmount(uint256 _amount) external onlyOwner {}

    function setCustodyWallet(address _wallet) external onlyOwner {}

    function setUnderlyingCap(uint256 _cap) external onlyOwner {}
```

4.In the BitFiStablecoin contract, the `Owner` role has the authority to set key contract parameters, set minters and their minting limits, set redeemers, and upgrade the contract through the UUPS upgrade mechanism. And the `Minter` and `Redeemer` roles have the authority to mint and burn tokens.

- contracts/bfusd/BitFiStablecoin.sol#L53, L57-L59, L61-L63, L65-L67, L69-L73, L75-L77

```
    function _authorizeUpgrade(address newImplementation) internal override onlyOwner
{}

    function setMinter(address _minter, uint256 _limit) external onlyOwner {}

    function increaseMinterLimit(address _minter, uint256 _amount) external onlyOwner
{}

    function setRedeemer(address _redeemer, bool _isActive) external onlyOwner {}

    function setManager(address _manager) external onlyOwner {}

    function setCrossChainFee(FeeLibrary.Fees calldata _fees) external onlyOwner {}

    function mint(address _to, uint256 _amount) external {}
```

```
    function burn(address _from, uint256 _amount) external onlyRedeemer {}
```

5.In the BitFiStablecoinManager contract, the `Owner` role has the authority to configure key contract parameters, control the address blacklist, control the fee whitelist, and upgrade the contract through the UUPS upgrade mechanism.

- contracts/bfusd/BitFiStablecoinManager.sol#L78, L163-L166, L168-L170, L172-L175, L177-L181, L183-L186, L188-L192, L194-L196

```
    function _authorizeUpgrade(address newImplementation) internal override onlyOwner
{}

    function setMultisig(address _newMultisig) external onlyOwner {}

    function setMinEpochPeriod(uint256 _minEpochPeriod) external onlyOwner {}

    function setFeeReceiver(address _newFeeReceiver) external onlyOwner {}

    function setFeeWhitelist(address _user, bool _isWhitelisted) external onlyOwner
{}

    function setStakingPool(address _pool, bool _isActive) external onlyOwner {}

    function setBlacklisted(address _account, bool _isBlacklisted) external onlyOwner
{}

    function setMerkleStash(address _stash) external onlyOwner {}
```

6.In the StakedBitFiStablecoin contract, the `Owner` role has the authority to configure key parameters of the contract and upgrade the contract through the UUPS upgrade mechanism.

- contracts/bfusd/staking/StakedBitFiStablecoin.sol#L125, L396-L398, L400-L402, L404-L406, L408-L412

```
    function _authorizeUpgrade(address newImplementation) internal override onlyOwner
{}

    function setFees(FeeLibrary.Fees calldata _fees) external onlyOwner {}

    function setCrossChainFee(FeeLibrary.Fees calldata _fees) external onlyOwner {}

    function setSettlementDelay(uint256 _delay) external onlyOwner {}
```

```
    function setDepositCap(uint256 _cap) external onlyOwner {}
```

7.In the BitFiStablecoinStandardRedeemer and the BitFiStablecoinInstantRedeemer contract, the `Owner` role has the authority to configure important contract parameters and upgrade the contract through the UUPS upgrade mechanism.

- contracts/bfusd/redeemers/BitFiStablecoinStandardRedeemer.sol#L83, L165-L167, L169-L171, L173-L175, L177-L179

```
    function _authorizeUpgrade(address newImplementation) internal override onlyOwner
{}

    function setFees(FeeLibrary.Fees calldata _fees) external onlyOwner {}

    function setSettlementDelay(uint256 _delay) external onlyOwner {}

    function setMinRedeemAmount(uint256 _amount) external onlyOwner {}

    function setCustody(address _custody) external onlyOwner {}
```

- contracts/bfusd/redeemers/BitFiStablecoinInstantRedeemer.sol#L72, L128-L130, L132-L134, L136-L138, L140-L142

```
    function _authorizeUpgrade(address newImplementation) internal override onlyOwner
{}

    function setFees(FeeLibrary.Fees calldata _fees) external onlyOwner {}

    function setStalePriceDelay(uint256 _delay) external onlyOwner {}

    function setMinRedeemAmount(uint256 _amount) external onlyOwner {}

    function setCustody(address _custody) external onlyOwner {}
```

Solution

In the short term, transferring owner ownership to multisig contracts is an effective solution to avoid single-point risk. But in the long run, it is a more reasonable solution to implement a privilege separation strategy and set up multiple privileged roles to manage each privileged function separately. And the authority involving user funds should be

managed by the community, and the EOA address can manage the authority involving emergency contract suspension. This ensures both a quick response to threats and the safety of user funds.

Status

Acknowledged

[N4] [Suggestion] Missing zero address check

Category: Others

Content

1.In the BfusdMerkleStash contract, the `initialize` and `setStakingPool` functions lack a zero-address check for the address type parameter.

- contracts/bfusd/BfusdMerkleStash.sol#L53-L61, L76-L78

```
    function initialize(address _owner, address _usd, address _manager, address
_pool) external initializer {
        //...
    }


    function setStakingPool(address _pool) external onlyOwner {
        //...
    }
```

2.In the BitFiStablecoinMinter contract, the `initialize` function lacks a zero-address check for the address type parameter.

- contracts/bfusd/BitFiStablecoinMinter.sol#L50-L74

```
    function initialize(
        address _owner,
        address _bfUSD,
        address _manager,
        address _underlying,
        address _priceFeed,
        address _custodyWallet,
        uint256 _minMintAmount,
        uint256 _initialCap
    ) external initializer {
```

```
        //...
    }
```

3.In the BitFiStablecoin contract, the `setManager` function lacks a zero-address check for the `_manager` parameter.

- contracts/bfusd/BitFiStablecoin.sol#L69-L73

```
    function setManager(address _manager) external onlyOwner {
        //...
    }
```

4.In the BitFiStablecoinManager contract, the `initialize` function lacks a zero-address check for the address type parameter.

- contracts/bfusd/BitFiStablecoinManager.sol#L60-L76

```
    function initialize(
        address _owner,
        IBitFiStablecoin _bfUSD,
        address _multisig,
        address _feeReceiver,
        uint256 _minEpochPeriod
    ) external initializer {
        //...
    }
```

5.In the StakedBitFiStablecoin contract, the `initialize` function lacks a zero-address check for the address type parameter.

- contracts/bfusd/staking/StakedBitFiStablecoin.sol#L91-L119

```
    function initialize(
        address _owner,
        IBitFiStablecoin _asset,
        IBitFiStablecoinManager _manager,
        string calldata _name,
        string calldata _symbol,
        uint256 _settlementDelay,
        uint256 _supplyCap,
        uint64 _initialRatio
    ) external initializer {
```

```
        //...
    }
```

6.In the BitFiStablecoinStandardRedeemer and BitFiStablecoinInstantRedeemer contract, the `initialize` function and `setCustody` function lack a zero-address check for the address type parameter.

- contracts/bfusd/redeemers/BitFiStablecoinStandardRedeemer.sol#L62-L81, L177-L179
- contracts/bfusd/redeemers/BitFiStablecoinInstantRedeemer.sol#L49-L70, L140-L142

```
    function initialize(
        address _owner,
        address _bfUSD,
        address _manager,
        address _underlying,
        address _custody,
        uint256 _settlementDelay,
        uint256 _minRedeemAmount
    ) external initializer {
        //...
    }

    function setCustody(address _custody) external onlyOwner {
        //...
    }
```

Solution

It is recommended to add zero address check.

Status

Acknowledged

[N5] [Suggestion] Missing event log

Category: Others

Content

1.In the BfusdMerkleStash contract, the `setStakingPool` function modifies important variables, but lacks event logging.

- contracts/bfusd/BfusdMerkleStash.sol#L76-L78

```
    function setStakingPool(address _pool) external onlyOwner {
        //...
    }
```

2.In the BitFiStablecoin contract, the functions `setMinter`, `increaseMinterLimit`, `setRedeemer`, `setManager`, and `setCrossChainFee` can modify permission configurations and key contract parameters, but they lack event logging.

- contracts/bfusd/BitFiStablecoin.sol#L57-L59, L61-L63, L65-L67, L69-L73, L75-L77

```
    function setMinter(address _minter, uint256 _limit) external onlyOwner {
        //...
    }

    function increaseMinterLimit(address _minter, uint256 _amount) external onlyOwner
{
        //...
    }

    function setRedeemer(address _redeemer, bool _isActive) external onlyOwner {
        //...
    }

    function setManager(address _manager) external onlyOwner {
        //...
    }

    function setCrossChainFee(FeeLibrary.Fees calldata _fees) external onlyOwner {
        //...
    }
```

3.In the BitFiStablecoinManager contract, the functions `setMultisig`, `setMinEpochPeriod`, `setFeeReceiver`, `setStakingPool`, and `setMerkleStash` can modify key parameters in the contract, but they lack event logging.

- contracts/bfusd/BitFiStablecoinManager.sol#L163-L166, L168-L170, L172-L175, L183-L186, L194-L196

```
    function setMultisig(address _newMultisig) external onlyOwner {
        //...
    }

    function setMinEpochPeriod(uint256 _minEpochPeriod) external onlyOwner {
        //...
```

```
    }

    function setFeeReceiver(address _newFeeReceiver) external onlyOwner {
        //...
    }

    function setStakingPool(address _pool, bool _isActive) external onlyOwner {
        //...
    }

    function setMerkleStash(address _stash) external onlyOwner {
        //...
    }
```

4.In the StoughtBitFiStablecoin contract, the functions `setFees`, `setCrossChainFee` and `setSettlementDelay` can modify important parameters in the contract, but they lack event logging.

- contracts/bfusd/staking/StakedBitFiStablecoin.sol#L396-L398, L400-L402, L404-L406, L408-L412

```
    function setFees(FeeLibrary.Fees calldata _fees) external onlyOwner {
        //...
    }

    function setCrossChainFee(FeeLibrary.Fees calldata _fees) external onlyOwner {
        //...
    }

    function setSettlementDelay(uint256 _delay) external onlyOwner {
        //...
    }
```

5.In the BitFiStablecoinStandardRedeemer the contract, the functions `setFees`, `setSettlementDelay`, `setMinRedeemAmount`, and `setCustody` can modify important parameters in the contract, but they lack event logging.

- contracts/bfusd/redeemers/BitFiStablecoinStandardRedeemer.sol#L165-L167, L169-L171, L173-L175, L177-L179

```
    function setFees(FeeLibrary.Fees calldata _fees) external onlyOwner {
        //...
    }

    function setSettlementDelay(uint256 _delay) external onlyOwner {
```

```
        //...
    }

    function setMinRedeemAmount(uint256 _amount) external onlyOwner {
        //...
    }

    function setCustody(address _custody) external onlyOwner {
        //...
    }
```

6.In the BitFiStablecoinInstantRedeemer the contract, the functions `setFees`, `setStalePriceDelay`, `setMinRedeemAmount`, and `setCustody` can modify important parameters in the contract, but they lack event logging.

- contracts/bfusd/redeemers/BitFiStablecoinInstantRedeemer.sol#L128-L130, L132-L134, L136-L138, L140-L142

```
    function setFees(FeeLibrary.Fees calldata _fees) external onlyOwner {}

    function setStalePriceDelay(uint256 _delay) external onlyOwner {}

    function setMinRedeemAmount(uint256 _amount) external onlyOwner {}

    function setCustody(address _custody) external onlyOwner {}
```

Solution

It is recommended to add event logging.

Status

Acknowledged

[N6] [Suggestion] Ignore function return value

Category: Others

Content

1.In the BfusdMerkleStash contract, the `claim` function ignores the return values of `usd.approve()` and `stakingPool.deposit()`.

- contracts/bfusd/BfusdMerkleStash.sol#L82-L105

```
    function claim(Claim[] calldata _claims) external whenNotPaused {
        //...
        if (totalAmount > 0) {
            usd.approve(address(stakingPool), totalAmount);
            stakingPool.deposit(totalAmount, msg.sender);
        }
    }
```

2.In the BitFiStablecoinZap contract, the `stake` function ignores the return value of `minter.bfUSD().approve()`.

- contracts/bfusd/BitFiStablecoinZap.sol#L59-L78

```
    function stake(
        IBitFiStablecoinMinter minter,
        IStakedBitFiStablecoin stakedBfusd,
        uint256 stableAmount,
        uint256 minOutput,
        address receiver
    ) external {
        //...
        minter.bfUSD().approve(address(stakedBfusd), bfusdAmount);
        //...
    }
```

3.In the StakedBitFiStablecoin contract, the `withdraw` and `redeem` functions ignore the return value of the `_withdraw` function.

- contracts/bfusd/staking/StakedBitFiStablecoin.sol#L154-L158, L160-L164

```
    function withdraw(uint256 assets, address receiver, address owner) public
override returns (uint256) {
        //...
        _withdraw(msg.sender, owner, receiver, shares, assets, epoch);
        //...
    }

    function redeem(uint256 shares, address receiver, address owner) public override
returns (uint256) {
        //...
        _withdraw(msg.sender, owner, receiver, shares, assets, epoch);
        //...
    }
```

4.In the StakedBitFiStablecoin contract, the `_claimWithdrawals` and `_debit` functions ignore the return value of `underlying.transfer()`.

- contracts/bfusd/staking/StakedBitFiStablecoin.sol#L300-L343, L442-L468

```
    function _claimWithdrawals(address user, uint256[] calldata withdrawalIds)
internal {
        //...
        if (aggregateFee > 0) {
            underlying.transfer(manager.feeReceiver(), aggregateFee);
        }
        if (totalClaimAssets > 0) {
            underlying.transfer(user, totalClaimAssets);
        }
    }

    function _debit(
        address _from,
        uint256 _amountLD,
        uint256 _minAmountLD,
        uint32 // dstEid
    ) internal virtual override returns (uint256 amountSentLD, uint256
amountReceivedLD) {
        //...
        if (fee > 0) {
            uint256 feeBfusd = (fee * currentRatio()) / RATIO_PRECISION;
            if (feeBfusd > 0) {
                underlying.transfer(manager.feeReceiver(), feeBfusd);
            }
        }
        //...
    }
```

5.In the BitFiStablecoinStandardRedeemer contract, the `requestRedemption` function ignores the return value of `bfUSD.transferFrom()`.

- contracts/bfusd/redeemers/BitFiStablecoinStandardRedeemer.sol#L87-L108

```
    function requestRedemption(uint256 _bfUSDAmount) external whenNotPaused returns
(uint256 redemptionId) {
        //...
        bfUSD.transferFrom(msg.sender, address(this), principalAmount);
```

```
        //...
    }
```

Solution

It is recommended to check or handle the function's return value.

Status

Acknowledged

[N7] [Suggestion] Inaccurate error type

Category: Design Logic Audit

Content

The `mint` and `_calculateMint` functions of the BitFiStablecoinMinter contract, and the `redeem` and `_calculateRedemption` functions of the BitFiStablecoinInstantRedeemer contract, all return the same `AmountTooLow` error, making it impossible for users to distinguish whether the transaction failed because the entered amount was lower than the protocol limit or because price fluctuations triggered the slippage protection mechanism.

- contracts/bfusd/BitFiStablecoinMinter.sol#LL80-L98, L111-L127

```
    function mint(uint256 _underlyingAmount, address _to, uint256 _minBfUSDAmount)
external whenNotPaused returns (uint256) {
        //...
        if (bfUSDAmount < _minBfUSDAmount) revert AmountTooLow();
        //...
    }

    function _calculateMint(
        uint256 _underlyingAmount,
        address _user
    ) internal view returns (uint256 bfUSDAmount, uint256 feeAmount, uint256
principalAmount) {
        if (_underlyingAmount < minMintAmount) revert AmountTooLow();
        //...
    }
```

- contracts/bfusd/redeemers/BitFiStablecoinInstantRedeemer.sol#L79-L95, L108-L124

```
    function redeem(uint256 _bfUSDAmount, address _to, uint256 _minUnderlyingAmount)
external whenNotPaused {
        //...
        if (underlyingToReceive < _minUnderlyingAmount) revert AmountTooLow();
        //...
    }

    function _calculateRedemption(
        uint256 _bfUSDAmount,
        address _user
    ) internal view returns (uint256 underlyingAmount, uint256 feeAmount, uint256
principalAmount) {
        if (_bfUSDAmount < minRedeemAmount) revert AmountTooLow();
        //...
    }
```

Solution

It is recommended to introduce a new slippage check for errors in the contract.

Status

Acknowledged

[N8] [Suggestion] Redundant code

Category: Others

Content

In the BitFiStablecoin contract, the `_calculateDebitAmounts` function repeatedly calculates the cross-chain transaction fees.

- contracts/bfusd/BitFiStablecoin.sol#L81-L95

```
    function _calculateDebitAmounts(address from, uint256 amount, uint256 minAmount)
internal view returns (uint256 fee) {
        //...
        uint256 feeAmount = FeeLibrary.calculateFee(amount, crossChainFee);
        if (feeAmount > 0) {
            fee = (amount * crossChainFee.percentageFee) / FeeLibrary.FEE_PRECISION;
            fee += crossChainFee.fixedFee;
        }
        //...
    }
```

Solution

It is recommended to remove redundant code.

Status

Acknowledged

5 Audit Result

Audit Number	Audit Team	Audit Date	Audit Result
0X002512050001	SlowMist Security Team	2025.12.01 - 2025.12.05	Medium Risk

Summary conclusion: The SlowMist security team use a manual and SlowMist team's analysis tool to audit the project, during the audit work we found 2 medium risk, 5 suggestion, 1 information.

6 Statement

SlowMist issues this report with reference to the facts that have occurred or existed before the issuance of this report, and only assumes corresponding responsibility based on these.

For the facts that occurred or existed after the issuance, SlowMist is not able to judge the security status of this project, and is not responsible for them. The security audit analysis and other contents of this report are based on the documents and materials provided to SlowMist by the information provider till the date of the insurance report (referred to as "provided information"). SlowMist assumes: The information provided is not missing, tampered with, deleted or concealed. If the information provided is missing, tampered with, deleted, concealed, or inconsistent with the actual situation, the SlowMist shall not be liable for any loss or adverse effect resulting therefrom. SlowMist only conducts the agreed security audit on the security situation of the project and issues this report. SlowMist is not responsible for the background and other conditions of the project.

Official Website

www.slowmist.com

E-mail

team@slowmist.com

Twitter

@SlowMist_Team

Github

https://github.com/slowmist