



## Audit Report for Fluid Finance - August 26, 2021

### Summary

FluidFi is a financial services firm based in Switzerland. The Audit Report prepared by Solidified covers smart contracts created by FluidFi in relation to its planned stablecoin, called fiatUSD. FluidFi intends to create real world assets on-chain, starting with US dollars held in its bank accounts. FluidFi is in the process of moving selected functions, including treasury, mint, and oracle functions to community governance for greater decentralization, in the form of the DeFi Bridge DAO.

### Process and Delivery

Three (3) independent Solidified experts performed an unbiased and isolated audit of the code below. The final debrief took place on August 26, 2021, and the results are presented here.

### Audited Files

The source code has been supplied in a private source code repository:

<https://gitlab.com/fluidfi/defi/stablecoin/-/tree/solidified-security-audit>

Commit number: `1cc2805e355d0b4a52df64c8498ebc193674ec46`

UPDATE: Fixes received on August 31 in MR:

[https://gitlab.com/fluidfi/defi/stablecoin/-/merge\\_requests/4/](https://gitlab.com/fluidfi/defi/stablecoin/-/merge_requests/4/)

Final commit number: `59e4e5a56b206140f884285e703cd78a3d1f542d`



## Audit Report for Fluid Finance - August 26, 2021

Audited file list:

- |— RealWorldAssetTether.sol
- |— oracles
  - |— CrossChainTotalSupplyOracle.sol
  - |— OffchainTreasuryOracle.sol

### Intended Behavior

It is intended that real world assets held by FluidFi (for example, US dollar account balances) will be accurately and transparently represented on-chain.

## Findings

Smart contract audits are an important step to improve the security of smart contracts and can find many issues. However, auditing complex codebases has its limits and a remaining risk is present (see disclaimer).

Users of a smart contract system should exercise caution. In order to help with the evaluation of the remaining risk, we provide a measure of the following key indicators: **code complexity**, **code readability**, **level of documentation**, and **test coverage**.

Note that high complexity or lower test coverage does not necessarily equate to a higher risk, although certain bugs are more easily detected in unit testing than a security audit and vice versa.

Criteria	Status	Comment
Code complexity	Low	-
Code readability and clarity	High	-
Level of Documentation	High	-
Test Coverage	High	-

## Issues Found

---

Solidified found that the Fluid Finance contracts contain no critical issues, no major issues, 1 minor issue, 4 informational notes, and 1 warning.

We recommend that issues are amended, while informational notes are up to the team's discretion, as they refer to best practices.

Issue #	Description	Severity	Status
1	RealWorldAssetTether.sol: Pending redeems can be released while the contract is paused	Minor	Resolved
2	RealWorldAssetTether.sol: oracleFeedStaleAfterTime can be set to unrealistically small amounts	Note	Resolved
3	RealWorldAssetTether.sol: Function updateFeeReceiver() does not validate _feeReceiver	Note	Resolved
4	CrossChainTotalSupplyOracle.sol: Unassigned role - Oracle updater	Note	Acknowledged
5	Misc Notes	Note	Acknowledged
6	Centralized oracles	Warning	Acknowledged

## Critical Issues

---

No critical issues have been found.

## Major Issues

---

No major issues have been found.

## Minor Issues

---

### 1. RealWorldAssetTether.sol: Pending redeems can be released while the contract is paused

---

Function `releasePendingRedeem()` does not check if the contract is paused before releasing the pending redeems.

#### Recommendation

Add the `whenNotPaused` modifier to `releasePendingRedeem()`.

#### Status

Resolved

## Informational Notes

---

### 2. RealWorldAssetTether.sol: oracleFeedStaleAfterTime can be set to unrealistically small amounts

---

`oracleFeedStaleAfterTime` can be set to unrealistically small amounts, which if set incorrectly by the `ORACLE_UPDATER_ROLE`, can cause all token transfers to revert until fixed.

#### Recommendation

Consider enforcing a minimum value (e.g. 1 day) for `oracleFeedStaleAfterTime` in `setOracleFeedStaleAfterTime()`.

#### Status

Resolved

### 3. RealWorldAssetTether.sol: Function updateFeeReceiver() does not validate `_feeReceiver`

---

Consider enforcing that `_feeReceiver != address(0)` in `updateFeeReceiver()`.

#### Status

Resolved

## 4. **CrossChainTotalSupplyOracle.sol: Unassigned role - Oracle updater**

---

The contract does not assign any user to the role **ORACLE\_UPDATER\_ROLE** by default.

### **Recommendation**

Consider explicitly specifying the role assignment during contract creation.

### **Status**

Acknowledged. Team's response: *"This is intentional, the ORACLE\_UPDATER\_ROLE will be granted explicitly for the respective account(s) in a subsequent step, not during deployment"*.

## 5. Misc Notes

---

- **CrossChainTotalSupplyOracle.sol**: Unused variable **snapshotBlockNrByChainID**. Consider removing unused variable. **Status:** Acknowledged. Team's response: *"This is intentional and an offchain requirement of the bridge process. We need to know which block heights are included in the latest crossChainTotalSupply update so we know which bridge transactions we can complete"*.

## Warnings

---

### 6. Centralized oracles

---

The oracle functions and the respective roles assigned are allowed to mint and redeem tokens. Though the system is designed to trust these oracles, they can act as a central point of failure or attack. The trust on the system is completely dependent on the trust on the oracles.

## Recommendation

It is strongly recommended to use a multi sig wallet or highly trusted system to handle the transactions.

## Status

Acknowledged. Team's response:

*This is not entirely correct. While it is true that the ORACLE\_UPDATER\_ROLE of the OffchainTreasuryOracle has the power to call the update function update(...) onlyRole(ORACLE\_UPDATER\_ROLE) such a call will only result in an actual mint of new tokens if the ORACLE\_UPDATER\_ROLE can provide a valid signature of the update (including the amount and designated recipient), signed by the offchainSigner:*

```
// OffchainTreasuryOracle.sol line 130  
if (ECDSA.recover(_hashSignedData(ui.amount, ui.account, ui.cefiTxID, ui.chainID),  
ui.signature) != offchainSigner) {  
    emit TxInvalidSignature(ui.cefiTxID);  
    continue;  
}
```

*The offchainSigner is controlled by the CeFi partner and custodian of the offchain treasury backing the RWAT token.*

*The ORACLE\_UPDATER\_ROLE will be controlled by the DeFi Bridge DAO that monitors the offchain treasury and onchain contracts and keeps onchain and offchain parties in sync.*

*While we fully agree that this is not yet a perfectly decentralised solution (we are working on a distributed consensus for both the offchainSigner as well as the ORACLE\_UPDATER\_ROLE), we would ask you to reconsider your conclusion that this is a "fully centralized oracle that can mint and redeem tokens at will" and a "centralized point of failure or attack".*





Audit Report for Fluid Finance - August 26, 2021

## **Disclaimer**

Solidified audit is not a security warranty, investment advice, or an endorsement of Fluid Finance or its products. This audit does not provide a security or correctness guarantee of the audited smart contract. Securing smart contracts is a multistep process, therefore running a bug bounty program as a complement to this audit is strongly recommended.

The individual audit reports are anonymized and combined during a debrief process, in order to provide an unbiased delivery and protect the auditors of Solidified platform from legal and financial liability.

*Solidified Technologies Inc.*