SOLIDIFIED

Audit Report for Reserve Protocol - June 6, 2024

Summary

Audit Report prepared by Solidified covering the Reserve Protocol 3.4.0 contracts.

Process and Delivery

Three (3) independent Solidified experts performed an unbiased and isolated audit of the code below. The final debrief took place on May 15, 2024, and the results are presented here.

Audited Files

The source code has been supplied in the following source code repository:

Repo: https://github.com/reserve-protocol/protocol/
Commit hash: Pull Request #1096 was reviewed at commit `d96e0cbf6ed57629ee64c2dbc6ae7159debaf509`

```
contracts
├── facade
│   ├── DeployerRegistry.sol
│   ├── Facade.sol
│   ├── FacadeMonitor.sol
│   ├── FacadeTest.sol
│   ├── FacadeWrite.sol
│   ├── facets
│   │   ├── ActFacet.sol
│   │   ├── MaxIssuableFacet.sol
│   │   └── ReadFacet.sol
│   └── lib
│       └── FacadeWriteLib.sol
├── interfaces
│   ├── IAsset.sol
│   ├── IAssetRegistry.sol
│   ├── IBackingManager.sol
│   ├── IBasketHandler.sol
│   ├── IBroker.sol
│   ├── IComponent.sol
│   ├── IDeployer.sol
```

```
│   ├── IDeployerRegistry.sol
│   ├── IDistributor.sol
│   ├── IFacade.sol
│   ├── IFacadeMonitor.sol
│   ├── IFacadeTest.sol
│   ├── IFacadeWrite.sol
│   ├── IFurnace.sol
│   ├── IGnosis.sol
│   ├── IMain.sol
│   ├── IRToken.sol
│   ├── IRTokenOracle.sol
│   ├── IRevenueTrader.sol
│   ├── IRewardable.sol
│   ├── IStRSR.sol
│   ├── IStRSRVotes.sol
│   ├── ITrade.sol
│   ├── ITrading.sol
│   └── IVersioned.sol
├── libraries
│   ├── Allowance.sol
│   ├── Array.sol
│   ├── Fixed.sol
│   ├── Permit.sol
│   ├── String.sol
│   └── Throttle.sol
├── mixins
│   ├── Auth.sol
│   ├── ComponentRegistry.sol
│   └── Versioned.sol
├── p0
│   ├── AssetRegistry.sol
│   ├── BackingManager.sol
│   ├── BasketHandler.sol
│   ├── Broker.sol
│   ├── Deployer.sol
│   ├── Distributor.sol
│   ├── Furnace.sol
│   ├── Main.sol
│   ├── RToken.sol
│   ├── RevenueTrader.sol
│   ├── StRSR.sol
```

```
|   └── mixins
|       ├── Component.sol
|       ├── Rewardable.sol
|       ├── Trading.sol
|       └── TradingLib.sol
├── p1
|   ├── AssetRegistry.sol
|   ├── BackingManager.sol
|   ├── BasketHandler.sol
|   ├── Broker.sol
|   ├── Deployer.sol
|   ├── Distributor.sol
|   ├── Furnace.sol
|   ├── Main.sol
|   ├── RToken.sol
|   ├── RevenueTrader.sol
|   ├── StRSR.sol
|   ├── StRSRVotes.sol
|   └── mixins
|       ├── BasketLib.sol
|       ├── Component.sol
|       ├── RecollateralizationLib.sol
|       ├── RewardableLib.sol
|       ├── TradeLib.sol
|       └── Trading.sol
├── plugins
|   ├── assets
|   |   ├── AppreciatingFiatCollateral.sol
|   |   ├── Asset.sol
|   |   ├── ERC4626FiatCollateral.sol
|   |   ├── EURFiatCollateral.sol
|   |   ├── FiatCollateral.sol
|   |   ├── L2LSDCollateral.sol
|   |   ├── NonFiatCollateral.sol
|   |   ├── OracleErrors.sol
|   |   ├── OracleLib.sol
|   |   ├── RTokenAsset.sol
|   |   ├── SelfReferentialCollateral.sol
|   |   ├── VersionedAsset.sol
|   |   ├── aave
|   |   |   ├── ATokenFiatCollateral.sol
```

```
│   │   │   ├── IStaticATokenLM.sol
│   │   │   ├── StaticATokenErrors.sol
│   │   │   ├── StaticATokenLM.sol
│   │   │   └── vendor
│   │   │       ├── ERC20.sol
│   │   │       ├── IAToken.sol
│   │   │       ├── IAaveIncentivesController.sol
│   │   │       ├── RayMathNoRounding.sol
│   │   │       └── ReentrancyGuard.sol
│   │   ├── aave-v3
│   │   │   ├── AaveV3FiatCollateral.sol
│   │   │   └── vendor
│   │   │       ├── ERC20.sol
│   │   │       ├── RayMathExplicitRounding.sol
│   │   │       ├── StaticATokenErrors.sol
│   │   │       ├── StaticATokenV3LM.sol
│   │   │       └── interfaces
│   │   │           ├── IAToken.sol
│   │   │           ├── IERC4626.sol
│   │   │           ├── IInitializableStaticATokenLM.sol
│   │   │           └── IStaticATokenV3LM.sol
│   │   ├── ankr
│   │   │   ├── AnkrStakedEthCollateral.sol
│   │   │   ├── README.md
│   │   │   └── vendor
│   │   │       └── IAnkrETH.sol
│   │   ├── cbeth
│   │   │   ├── CBETHCollateral.sol
│   │   │   ├── CBETHCollateralL2.sol
│   │   │   ├── README.md
│   │   │   └── vendor
│   │   │       └── ICBEth.sol
│   │   ├── compoundv2
│   │   │   ├── CTokenFiatCollateral.sol
│   │   │   ├── CTokenNonFiatCollateral.sol
│   │   │   ├── CTokenSelfReferentialCollateral.sol
│   │   │   ├── DEPRECATED_CTokenWrapper.sol
│   │   │   └── ICToken.sol
│   │   ├── compoundv3
│   │   │   ├── CTokenV3Collateral.sol
│   │   │   ├── CometHelpers.sol
```

```
│   │   │   ├── CusdcV3Wrapper.sol
│   │   │   ├── ICusdcV3Wrapper.sol
│   │   │   ├── IWrappedERC20.sol
│   │   │   ├── WrappedERC20.sol
│   │   │   └── vendor
│   │   │       ├── CometCore.sol
│   │   │       ├── CometExtInterface.sol
│   │   │       ├── CometExtMock.sol
│   │   │       ├── CometInterface.sol
│   │   │       ├── CometMainInterface.sol
│   │   │       ├── CometStorage.sol
│   │   │       ├── IComet.sol
│   │   │       ├── ICometConfigurator.sol
│   │   │       ├── ICometProxyAdmin.sol
│   │   │       └── ICometRewards.sol
│   │   ├── curve
│   │   │   ├── CurveAppreciatingRTokenFiatCollateral.sol
│   │   │   ├── CurveAppreciatingRTokenSelfReferentialCollateral.sol
│   │   │   ├── CurveRecursiveCollateral.sol
│   │   │   ├── CurveStableCollateral.sol
│   │   │   ├── CurveStableMetapoolCollateral.sol
│   │   │   ├── CurveStableRTokenMetapoolCollateral.sol
│   │   │   ├── PoolTokens.sol
│   │   │   ├── README.md
│   │   │   ├── crv
│   │   │   │   └── CurveGaugeWrapper.sol
│   │   │   ├── cvx
│   │   │   │   └── vendor
│   │   │   │       ├── ConvexInterfaces.sol
│   │   │   │       ├── ConvexStakingWrapper.sol
│   │   │   │       ├── CvxMining.sol
│   │   │   │       ├── IConvexStakingWrapper.sol
│   │   │   │       ├── IRewardStaking.sol
│   │   │   │       └── StableSwap3Pool.vy
│   │   │   └── stakedao
│   │   │       ├── IStakeDAO.sol
│   │   │       └── StakeDAORecursiveCollateral.sol
│   │   ├── dsr
│   │   │   ├── README.md
│   │   │   └── SDaiCollateral.sol
│   │   ├── erc20
```

```
|   |   |   ├── RewardableERC20.sol
|   |   |   ├── RewardableERC20Wrapper.sol
|   |   |   └── RewardableERC4626Vault.sol
|   |   ├── frax
|   |   |   ├── README.md
|   |   |   └── SFraxCollateral.sol
|   |   ├── frax-eth
|   |   |   ├── README.md
|   |   |   ├── SFraxEthCollateral.sol
|   |   |   └── vendor
|   |   |       ├── IfrxEthMinter.sol
|   |   |       └── IsfrxEth.sol
|   |   ├── lido
|   |   |   ├── L2LidoStakedEthCollateral.sol
|   |   |   ├── LidoStakedEthCollateral.sol
|   |   |   ├── README.md
|   |   |   └── vendor
|   |   |       ├── ISTETH.sol
|   |   |       └── IWSTETH.sol
|   |   ├── meta-morpho
|   |   |   ├── MetaMorphoFiatCollateral.sol
|   |   |   ├── MetaMorphoSelfReferentialCollateral.sol
|   |   |   └── README.md
|   |   ├── morpho-aave
|   |   |   ├── IMorpho.sol
|   |   |   ├── MorphoAaveV2TokenisedDeposit.sol
|   |   |   ├── MorphoFiatCollateral.sol
|   |   |   ├── MorphoNonFiatCollateral.sol
|   |   |   ├── MorphoSelfReferentialCollateral.sol
|   |   |   ├── MorphoTokenisedDeposit.sol
|   |   |   └── README.md
|   |   ├── rocket-eth
|   |   |   ├── README.md
|   |   |   ├── RethCollateral.sol
|   |   |   └── vendor
|   |   |       ├── IReth.sol
|   |   |       ├── IRocketNetworkBalances.sol
|   |   |       └── IRocketStorage.sol
|   |   ├── stargate
|   |   |   ├── DO_NOT_USE_StargatePoolETHCollateral.sol
|   |   |   ├── README.md
```

```
│   │   │   ├── StargatePoolFiatCollateral.sol
│   │   │   ├── StargateRewardableWrapper.sol
│   │   │   └── interfaces
│   │   │       ├── IStargateLPStaking.sol
│   │   │       ├── IStargatePool.sol
│   │   │       └── IStargateRouter.sol
│   │   └── yearnv2
│   │       ├── README.md
│   │       └── YearnV2CurveFiatCollateral.sol
│   ├── governance
│   │   └── Governance.sol
│   └── trading
│       ├── DutchTrade.sol
│       └── GnosisTrade.sol
└── vendor
    ├── ERC20PermitUpgradeable.sol
    └── oz
        ├── ERC4626.sol
        ├── IERC4626.sol
        └── Multicall.sol
```

Intended Behavior

The audited codebase implements a generic framework to issue tokens that are backed by a rebalancing basket of collateral.

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	Medium	-
Code readability and clarity	High	-
Level of Documentation	High	-
Test Coverage	High	Extensive test suite with almost 100% coverage. Whenever something is not covered, it is argued why.

Issues Found

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

Issue #	Description	Severity	Status
1	Dependence on external governance might make claimRewards() run out of gas	Minor	Acknowledged
2	No checks for Arbitrum Sequencer liveness	Minor	Acknowledged
3	Wrong naming of the CurveAppreciatingRTokenSelfReferentialCollateral contract	Note	Acknowledged
4	Important changes in CurveStableRTokenMetapoolCollateral left undocumented	Note	Acknowledged
5	Outdated comments due to a shift from block.number to block.timestamp	Note	Acknowledged
6	Miscellaneous	Note	Acknowledged
7	Known issues and limitations	Note	Acknowledged

Critical Issues

No critical issues have been found.

Major Issues

No major issues have been found.

Minor Issues

1. Dependence on external governance might make claimRewards() run out of gas

In StakeDAORecursiveCollateral.sol, iterations over the count of the rewards might run out of gas; this could fail the claimRewards() function.

This issue is minor as it can only be triggered by an external governance. This might not only result from a hostile attack but could also happen by accident, as the external governance is not guaranteed to be aware of this.

Recommendation
Implement a limit or pagination.

2. No checks for Arbitrum Sequencer liveness

Using ChainLink oracles in L2 chains such as Arbitrum requires checking if the sequencer is down to avoid prices from looking like they are fresh when they are not. The issue could be leveraged by malicious actors to take advantage of the sequencer downtime.

Recommendation
Consider checking whether the sequencer is up before consuming any price data. The Chainlink docs on L2 Sequencer Uptime Feeds specify more details.

Informational Notes

3. Wrong naming of the `CurveAppreciatingRTokenSelfReferentialCollateral` contract

In the current protocol design, setting `defaultThreshold` to `0` implies no deviation tolerance in either direction. As a result, if the market price shifts 1 wei from `FIX_ONE` a depeg is registered. Hence, for all self-referential collaterals, a `config.defaultThreshold == 0` check should be included.

However, the current `CurveAppreciatingRTokenSelfReferentialCollateral` implementation inherits from `CurveStableCollateral`, which has an opposite `config.defaultThreshold != 0` check.

Also, as stated in the contract comments `this plugin contract is intended for use with a CurveLP token for a pool between a self-referential reference token(WETH) and an RToken that is appreciating relative to it` which means it does not fulfill the requirement of {tok} == {ref}, {ref} == {target}, {target} != {UoA} to be named as a SelfReferentialCollateral.

Recommendation
Consider renaming `CurveAppreciatingRTokenSelfReferentialCollateral` or clarifying this.

4. Important changes in `CurveStableRTokenMetapoolCollateral` left undocumented

The `refresh` function of `CurveStableRTokenMetapoolCollateral` was updated to check the RToken status and, as a consequence, will default in case RToken accepts a devaluation. If this is an intended design, acknowledge this in a comment.

Recommendation
Add the necessary documentation. For instance, by adding the comment to the contract file. The following already exists in `CurveAppreciatingRTokenFiatCollateral`:

```
      * Warning: Defaults after haircut! After the RToken accepts a devaluation this collateral
      * plugin will default and the collateral will be removed from the basket.
```

5. Outdated comments due to a shift from `block.number` to `block.timestamp`

The protocol has moved from using `block.number` to `block.timestamp` in order to work with Arbitrum. However, some comments have not been updated accordingly, which might lead to confusion.

Recommendation
In the `Auth` contract, change the comment `Typically freezing thaws on its own in a predetermined number of blocks` to `Typically freezing thaws on its own in a predetermined time point`.
In the `IStRSRVotes` interface, change the comment from `/// @return The era at a past block number` to `/// @return The era at a past time point`.

6. Miscellaneous

The following are some recommendations to improve the code quality and readability:

- Unnecessary `IVotesUpgradeable` import in `IStRSRVotes` interface as it is already imported in inheriting `IERC5805Upgradeable`.

- Unnecessary `IERC5805Upgradeable` import in `StRSRP1Votes` contract as it is already imported in inheriting `IStRSRVotes`.
- Unnecessary `IERC20` import in IRToken interface as it is already imported in inheriting `IERC20MetadataUpgradeable`. Also, every time `IRToken` is imported in other contracts, it's unnecessary to import `IERC20` again.

7. Known issues and limitations

The protocol has acknowledged issues found in previous audit reports (see e.g. Audits performed by Solidified) and other known limitations and issues.

To maintain compatibility across various blockchain networks such as Ethereum and layer-2 solutions like Arbitrum, Reserve 3.4.0 utilizes the `block.timestamp` (block time) rather than the `block.number` to manage certain time-sensitive operations. This design choice is intended to avoid the need to maintain different versions of the contract tailored to each blockchain's characteristics. Using `block.timestamp` introduces a potential vector for minor manipulations by validators or miners. In certain conditions, these actors might be able to adjust the timestamp of the blocks they produce within reasonable limits, which could be exploited to derive miner extractable value (MEV).

Disclaimer

Oak Security GmbH