yAudit USA.d Review

Review Resources:

- Liquity documentation

Auditors:

- Panda
- Adriro

Table of Contents

{: .no_toc }

Review Summary

USA.d

USD.a is a synthetic dollar, part of a collateralized debt position (CDP), backed by a stablecoin, ETH, and BTC basket.

The codebase is forked from Liquity V2 (codename bold) and adjusted to include the collateral changes.

The contracts of the USA.d Repo were reviewed over five days. The code review was performed by two auditors between 27 January and 31 January 2025. The repository was under active development during the review, but the review was limited to the latest commit d74f1532535e6471bbe437d16616ffe12fc36ecf for the USA.d repo.

Scope

The scope of the review consisted of the following contracts at the specific commit:

```
contracts/script/DeployUSA.D.s.sol
contracts/src/PriceFeeds/MainnetPriceFeedBase.sol
contracts/src/PriceFeeds/USA.D/BaseOracle.sol
contracts/src/PriceFeeds/USA.D/ERC4626Oracle.sol
contracts/src/PriceFeeds/USA.D/Fallbacks/BaseFallbackOracle.sol
contracts/src/PriceFeeds/USA.D/Fallbacks/CrvUsdFallbackOracle.sol
contracts/src/PriceFeeds/USA.D/Fallbacks/SfrxEthFallbackOracle.sol
contracts/src/PriceFeeds/USA.D/Fallbacks/TbtcFallbackOracle.sol
contracts/src/PriceFeeds/USA.D/Fallbacks/WbtcFallbackOracle.sol
contracts/src/PriceFeeds/USA.D/ScrvUsdOracle.sol
contracts/src/PriceFeeds/USA.D/SdaiOracle.sol
contracts/src/PriceFeeds/USA.D/SfrxEthOracle.sol
contracts/src/PriceFeeds/USA.D/TbtcOracle.sol
contracts/src/PriceFeeds/USA.D/WbtcOracle.sol
```

```
contracts/src/TroveNFT.sol
contracts/src/Zappers/USAZapper.sol
```

After the findings were presented to the USA.d team, fixes were made and included in several PRs.

This review is a code review to identify potential vulnerabilities in the code. The reviewers did not investigate security practices or operational security and assumed that privileged accounts could be trusted. The reviewers did not evaluate the security of the code relative to a standard or specification. The review may not have identified all potential attack vectors or areas of vulnerability.

yAudit and the auditors make no warranties regarding the security of the code and do not warrant that the code is free from defects. yAudit and the auditors do not represent nor imply to third parties that the code has been audited nor that the code is free from defects. By deploying or using the code, USA.d and users of the contracts agree to use the code at their own risk.

Code Evaluation Matrix

Category	Mark	Description
Access Control	Good	Most changes don't require access control. MockInterestRouter and MetadataNFT functions are restricted to the Asymmetry multisig.
Mathematics	Low	One severe issue was found in the oracle math.
Complexity	Average	While the changes do not involve an increase in complexity, it is important to note that Liquity itself is a complex protocol..
Libraries	Good	Standard, well-audited libraries used. It relies on the OpenZeppelin library and the Uniswap V3 periphery contracts to interface with the oracles.
Decentralization	Good	The protocol inherits the decentralized nature of Liquity v2.
Code stability	Good	No changes were made to the contract during the review.
Documentation	Good	Being a Liquity fork, plenty of documentation is available.
Monitoring	Good	Sufficient event emissions for tracking operations.
Testing and verification	Average	More tests around oracles are suggested.

Findings Explanation

Findings are broken down into sections by their respective impact:

- Critical, High, Medium, Low impact
    - These are findings that range from attacks that may cause loss of funds, impact control/ownership of the contracts, or cause any unintended consequences/actions that are outside the scope of the requirements.
- Gas savings
    - Findings that can improve the gas efficiency of the contracts.
- Informational
    - Findings including recommendations and best practices.

Critical Findings

1. Critical - Incorrect price calculation in WBTC Oracle

The WBTC Oracle uses the WBTC/BTC and BTC/USD Chainlink price feeds to calculate the price. However, the underlying calculation is incorrect, as it divides the BTC/USD price by the WBTC/BTC price.

Technical Details

The WbtcOracle.sol implementation fetches both prices and divides them to get the WBTC/USD price.

```
59:         int256 wbtcUsdPrice = btcUsdPrice * int256(10 ** decimals()) /
wbtcBtcPrice;
```

Since the WBTC/BTC price represents the amount of WBTC in terms of BTC, and the BTC/USD represents the amount of BTC in terms of USD, the calculation should multiply both prices and adjust the scale.

Impact

Critical. The WBTC Oracle price is incorrect.

Recommendation

The calculation should multiply both prices and divide by the scale.

```
-   int256 wbtcUsdPrice = btcUsdPrice * int256(10 ** decimals()) / wbtcBtcPrice;
+   int256 wbtcUsdPrice = wbtcBtcPrice * btcUsdPrice / int256(1e8) ;
```

Developer Response

Fixed e7dffc2.

High Findings

None

Medium Findings

None

Low Findings

1. Low - Incorrect staleness threshold for sfrxETH Oracle

The Oracle is configured with a 24-hour threshold, while the underlying Chainlink Oracle uses one hour.

Technical Details

The SfrxEthOracle.sol contract is configured with a 24-hour staleness threshold in the deployment script.

```
323:         } else if (address(_collToken) == SFRXETH) {
324:             _stalenessThreshold = _24_HOURS; // CL FRAX/ETH heartbeat
(Fallback). Primary FRAX/USD is 1 hour, but falls back
325:             SfrxEthFallbackOracle fallbackOracle = new
SfrxEthFallbackOracle();
326:             contracts.oracle = address(new
SfrxEthOracle(address(fallbackOracle)));
327:             sfrxEthFallbackOracle = fallbackOracle;
```

The comment indicates the fallback uses a 24-hour threshold due to the FRAX/ETH Chainlink price feed. However, the fallback is based on Curve's Price Aggregator, which doesn't expose a last update timestamp.

Impact

Low.

Recommendation

Change the threshold to one hour.

Developer Response

Fixed f606f9f

2. Low - Do not set the same value for MCR and SCR

Technical Details

In the `DeployUSA.D.s.sol` script, the TroveManager parameters are set with equal MCR (Minimum Collateral Ratio) and SCR (System Critical Ratio) values for all collateral types:

```
troveManagerParamsArray[0] = TroveManagerParams(150e16, 110e16, 110e16, 5e16,
10e16); // scrvUSD
troveManagerParamsArray[1] = TroveManagerParams(150e16, 110e16, 110e16, 5e16,
10e16); // sDAI
troveManagerParamsArray[2] = TroveManagerParams(150e16, 110e16, 110e16, 5e16,
10e16); // sfrxETH
troveManagerParamsArray[3] = TroveManagerParams(150e16, 110e16, 110e16, 5e16,
10e16); // tBTC
troveManagerParamsArray[4] = TroveManagerParams(150e16, 110e16, 110e16, 5e16,
10e16); // WBTC
```

For each collateral type, both MCR and SCR are set to 110e16 (110%). When these values are equal, a single trove that becomes liquidatable (falls below MCR) can trigger a system-wide shutdown of that particular branch, as the total collateral ratio would fall below the SCR threshold.

If a branch has only one trove and that trove becomes liquidatable, the entire branch will be forced to shut down. This creates a vulnerability where a single borrower could intentionally or unintentionally cause a branch-wide shutdown, affecting all potential users of that collateral type.

Impact

Low.

Recommendation

There are two possible solutions:

`. Set the SCR lower than the MCR to create a buffer zone between when individual troves become liquidatable and when the system enters recovery mode. For example:

```
troveManagerParamsArray[0] = TroveManagerParams(150e16, 110e16, 105e16, 5e16,
10e16); // SCR < MCR
```

2. As part of the deployment process:

- Open a trove on each branch
- Redeem it down to 0 debt
- Keep it open for each branch

Developer Response

Acknowledged.

Gas Saving Findings

None

Informational Findings

1. Informational - Consider using sUSDS as a collateral

Sky.money has introduced a new stablecoin called USDS, which can be exchanged 1:1 with DAI.

Technical Details

The Total Value Locked (TVL) of Sky.money has grown rapidly, to the detriment of DAI. Currently, the TVL of USDS is bigger than that of DAI. As more funds transition to USDS, incorporating it as an additional collateral type will maintain the ability to use the Sky.money/MakerDAO stablecoin effectively.

As more funds are moving to USDS, having USDS as another collateral will keep the possibility of using the sky/MakerDao stable coin.

Impact

Informational.

Recommendation

Add sUSDS as a collateral

Developer Response

USDS added as a collateral 62e1ef

2. Informational - Incorrect constant name

Technical Details

The constants `CL_TBTC_BTC_PRICE_FEED` and `_CL_TBTC_BTC_HEARTBEAT` are incorrectly named as they suggest a tBTC/BTC price feed, but the address `0x8350b7De6a6a2C1368E7D4Bd968190e13E354297` actually corresponds to a tBTC/USD Chainlink price feed.

TbtcOracle.sol#L10-L12

Impact

Informational.

Recommendation

Rename the constants to reflect that it's an accurate USD price feed.

Developer Response

Fixed 97a971a

3. Informational - FallbackOracle price could rely on chainlink response.

The `SfrxEthFallbackOracle`, `TbtcFallbackOracle`, and `WbtcFallbackOracle` contracts rely on CryptoWithStablePriceSfrxeth, CryptoWithStablePriceTBTC and CryptoWithStablePriceWBTC. The three contracts from curve have the option to use a chainlink oracle as part of the `price()` function calculation. The call to the Chainlink Oracle is currently disabled but can be turned on.

Technical Details

We are going to focus on the `SfrxEthOracle` but this issue is similar with on the other oracles.

The `SfrxEthOracle` has a fallback oracle if the oracle response is zero. The fallback oracle uses the contract `CryptoWithStablePriceSfrxeth` this contract can use chainlink as an oracle. Both `CryptoWithStablePriceSfrxeth` and `SfrxEthOracle`. When enabled in `CryptoWithStablePriceSfrxeth`. If the `answer` value from the oracle is zero but the oracle is

properly updated, the likely wrong price from Chainlink will be used, and the fallback oracle will also return a zero value.

Impact

Informational

Recommendation

- Monitor if the `CryptoWithStablePriceSfrxeth` contract `use_chainlink` is set to true.
- Disable the fallback oracle if it is set to use chainlink

Developer Response

Will be monitored.

4. Informational - Replace assets for logos

The deployment script still references Liquity's assets.

Technical Details

The MetadataDeployment.sol contract loads the logos for the original Liquity assets (BOLD, WETH, rETH, wstETH, and geist).

Impact

Informational.

Recommendation

Replace the assets for the logos used in the protocol: USA.D, scrvUSD, sDAI, sfrxETH, tBTC, and WBTC.

Developer Response

5. Informational - Revert in unimplemented Zapper functions

There are several functions in USAZapper.sol, inherited from the BaseZapper interface, that are left unimplemented.

Technical Details

- `closeTroveFromCollateral()`
- `receiveFlashLoanOnCloseTroveFromCollateral()`

- `receiveFlashLoanOnOpenLeveragedTrove()`
- `receiveFlashLoanOnLeverUpTrove()`
- `receiveFlashLoanOnLeverDownTrove()`

Impact

Informational.

Recommendation

Consider reverting these functions to state that these shouldn't be called explicitly. Alternatively, the inherited interface could be adjusted to remove them.

Developer Response

Won't implement.

Final remarks

USA.d is a fork of Liquity with limited modifications built to create a stablecoin backed by BTC, ETH derivatives, and DAI/USDS collaterals. It contains new oracles relying on chainlink with fallback oracles in case of a failure to get a value from the chainlink oracles. The codebase is well documented, and the changes are isolated from the original codebase. The team has been responsive to the findings and has fixed the critical issue found during the review. The auditors recommend further testing around the oracles and the zappers.