yAudit Asymmetry USDaf V2 Review

Review Resources:

- Protocol documentation

Auditors:

- HHK
- Adriro

Table of Contents

Review Summary

USDaf

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

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

The contracts of the USDaf repository were reviewed over three days. Two auditors performed the code review between May 19th and May 21st, 2025. The repository was under active development during the review, but the review was limited to the latest commit cc3bd0537ad4555c34ab58ca285c11284c252fec.

Scope

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

```
contracts/script/DeployUSDaf.s.sol
contracts/src/BoldToken.sol
contracts/src/InterestRouter.sol
contracts/src/TroveNFT.sol
contracts/src/NFTMetadata/MetadataNFT.sol
contracts/src/PriceFeeds/USDaf/BaseOracle.sol
contracts/src/PriceFeeds/USDaf/CbbtcOracle.sol
contracts/src/PriceFeeds/USDaf/ERC4626Oracle.sol
contracts/src/PriceFeeds/USDaf/ScrvUsdOracle.sol
contracts/src/PriceFeeds/USDaf/SdaiOracle.sol
contracts/src/PriceFeeds/USDaf/SfrxUsdOracle.sol
contracts/src/PriceFeeds/USDaf/SusdeOracle.sol
contracts/src/PriceFeeds/USDaf/SusdsOracle.sol
contracts/src/PriceFeeds/USDaf/TbtcOracle.sol
contracts/src/PriceFeeds/USDaf/WbtcOracle.sol
contracts/src/PriceFeeds/USDaf/Fallbacks/BaseFallbackOracle.sol
contracts/src/PriceFeeds/USDaf/Fallbacks/CbbtcFallbackOracle.sol
contracts/src/PriceFeeds/USDaf/Fallbacks/CrvUsdFallbackOracle.sol
contracts/src/PriceFeeds/USDaf/Fallbacks/TbtcFallbackOracle.sol
contracts/src/PriceFeeds/USDaf/Fallbacks/WbtcFallbackOracle.sol
contracts/src/WrappedCbbtc.sol
contracts/src/WrappedWbtc.sol
contracts/src/Zappers/CbbtcZapper.sol
```

```
contracts/src/Zappers/WbtcZapper.sol
contracts/src/Zappers/ZapperAsFuck.sol
```

After the findings were presented to the Asymmetry 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, Asymmetry 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. The InterestRouter contract, owned by the Asymmetry multisig, acts as the Liquity Governance replacement.
Mathematics	Good	No arithmetic issues were detected.
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	The codebase relies on the OpenZeppelin library.
Decentralization	Good	The protocol inherits the decentralized nature of Liquity.
Code stability	Good	No changes were made to the contract during the review.
Documentation	Good	Being a Liquity fork, there is plenty of documentation available.
Monitoring	Good	Monitoring events are correctly emitted.
Testing and verification	Average	Even though Liquity has been extensively tested, additional testing of the zapper and pricing functionality is recommended.

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

None.

High Findings

None.

Medium Findings

None.

Low Findings

1. Low - Possible hard and soft redemptions frontrunning

The newly added oracles don't inherit the redemption frontrunning protection implemented by Liquity on their rETH and stETH oracles.

Technical Details

The Liquity documentation describes a potential issue known as Oracle price frontrunning. In this section, they describe how redemptions could be front-run because the Oracle update thresholds for rETH and stETH are 2%, much higher than the original ETH-USD price feed set to 0.5%.

The document describes a hard frontrunning attack sequence as this:

- Observe collateral price increase Oracle update in the mempool
- Frontrun with redemption transaction for $xworthofBOLDandreceive$x - fee worth of collateral
- Oracle price increase update transaction is validated
- Sell redeemed collateral for $ysuchthat$y > $x, due to the price increase
- Extracts $(y-x) from the system.

And a soft frontrunning as this:

When the attacker sees the market price increase before even seeing the Oracle price update transaction in the mempool.

In both cases, the value extracted is higher than what is actually needed to restore the peg. It can even be performed above the peg.

Some new oracles have similar update thresholds; this is the case for wBTC, cbBTC, and tBTC.

Excessive redemptions of these collaterals could negatively impact growth and user deposits.

Impact

Low. Users may be able to perform excessive redemptions.

Recommendation

Implement a similar logic as the Liquity rETH and stETH oracles, see here.

- Add canonical pricing. Since these are supposed to be 1-1 backed by BTC, you could use the BTC-USD price feed and assume 1-1 backing.
- Check if the price request is for redemption.
- Check if the derivation is below 2%; if true, use the max between the canonical and original price feed.
- If the derivation is above 2% or not in a redemption, use the minimum between the canonical and original price feed.

Developer Response

Fixed in 5724f27.

Gas Saving Findings

1. Gas - Fallback logic is not used in sUSDe Oracle

Technical Details

No fallback Oracle is configured for the sUSDe Oracle, so the fallback mechanism cannot be triggered.

Impact

Gas savings.

Recommendation

Remove the fallback check. Since the implementation only forwards the underlying Chainlink response, the sUSDe price feed can be directly configured using the Chainlink address.

Developer Response

Fixed in d6008d4.

Informational Findings

1. Informational - Cleanup the deployment script for better readability

Technical Details

The deployment script has a lot of unused code that could be removed to make it easier to read and safer to use.

- Lines 121 to 128 are not used and can be removed.
- Lines 139 to 174 are not used and can be removed.
- `r.usdcCurvePool` and the comment above can be renamed into curveStrategicPool.
- Lines 393 to 403 are commented out and can be removed.
- Param `_usdcCurvePool` of the function `_deployAndConnectCollateralContracts()` is not used and can be removed.
- Param `_otherToken` of the function `_deployCurvePool()` is not used and can be removed.

Impact

Informational.

Recommendation

Clean up the deployment script from unused code.

Developer Response

Partially fixed in a61d0e5.

2. Informational - Upstream changes in Zapper

Some changes in Liquity's zappers can be adapted to the USDaf zapper.

Technical Details

- Include the original caller address in the index when opening a trove:
  - WETHZapper.sol#L32
- Ensure the zapper contract is the actual receiver when redeeming assets:
  - WETHZapper.sol#L92
  - WETHZapper.sol#L105
  - WETHZapper.sol#L231

Impact

Informational.

Recommendation

Port the new changes to the USDaf zapper contract.

Developer Response

Fixed in 65d46f5, 7048c4b and 3487f56.

3. Informational - Use USDS Chainlink price feed instead of DAI in the USDS oracle

Technical Details

The USDS oracle currently uses the DAI Chainlink price feed.

USDS and DAI are convertible, and some existing lending markets, such as AAVE and Morpho, use DAI price feeds to price USDS.

However, given Liquity's immutable nature, if the DAI to USDS convertibility were to change, it could become dangerous for the protocol to keep using the DAI price feed.

For this reason, it seems safer to use the appropriate USDS chainlink price feed. Some existing lending markets, such as Compound and Euler, currently use it, demonstrating its robustness.

Impact

Informational.

Recommendation

Use the USDS price feed in the USDS oracle and update the heartbeat.

Because this feed has a 2% threshold, you may want to implement behavior similar to Low#1 for redemptions.

Developer Response

Acknowledged. We will use the DAI oracle.

Final Remarks

The new and updated version of Asymmetry's USDaf contains upstream changes and fixes from Liquity, along with changes to the selected collaterals: sfrxETH was removed. At the same time, sfrxUSD, sUSDe, and cbBTC were introduced.

A potential oracle front-running issue in the BTC collaterals was reported due to the reliance on high-threshold (2%) Chainlink feeds. This was resolved as recommended by strengthening the BTC price feed implementation, which is similar to what Liquity uses for wstETH or rETH.