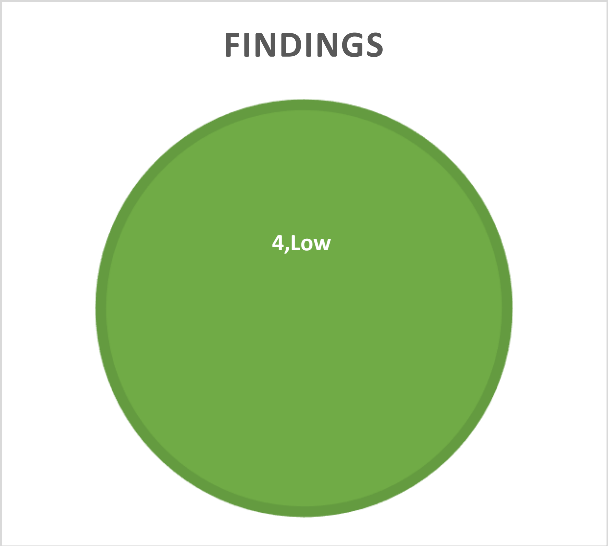
Trust Security

Smart Contract Audit

Reserve Protocol –USDM, sUSDe, apxETH

11/07/24

Executive summary

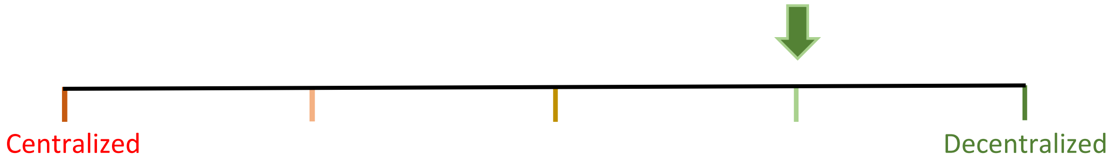


Category	Stablecoin
Auditor	HollaDieWaldfee

Findings

Severity	Total	Fixed	Open	Acknowledged
High	-	-	-	-
Medium	-	-	-	-
Low	4	3	-	1

Centralization score



Signature

Digitally signed by Trust
Date: 2024.07.11 15:43:59 +03'00'

Document properties

Versioning

Version	Date	Description
0.1	11/07/2024	Client report

Contact

Trust

trust@trust-security.xyz

Introduction

Trust Security serves as a long-term security partner of the Reserve Protocol. It has conducted the audit at the customer's request. The audit is focused on uncovering security issues and additional bugs contained in the code defined in scope. Additional recommendations have been given when appropriate.

Scope

The following PRs are in scope of the audit:

- USDM plugin
- sUSDe plugin
- apxETH plugin

Repository details

- **Repository URL:** https://github.com/reserve-protocol/protocol
- **Commit hash (USDM):** 01dc973e460cb70f1d2ff2d5f52d2207218b3045
- **Commit hash (sUSDe):** 1e4ea3951d3fb9395ca33ab0bbd3074a2a96ba2c
- **Commit hash (apxETH):** 98ebac8165df7ccd9bf721b8fcdf2172c7338ab1
- **Mitigation hash (USDM):** 6dab583380d2548197b649a4cbab3b6050c0c3cd
- **Mitigation hash (sUSDe):** c2c868710d6d39490f1e4b624d73868b8d19fffc
- **Mitigation hash (apxETH):** 83f8823e9c460a27b505544bc67f7888b4a974aa

About Trust Security

Trust Security has been established by top-end blockchain security researcher Trust, in order to provide high quality auditing services. Trust is the leading auditor at competitive auditing service Code4rena, reported several critical issues to Immunefi bug bounty platform and is currently a Code4rena judge.

About the Auditors

HollaDieWaldfee is a renowned security expert with a track record of multiple first places in competitive audits. He is a Lead Senior Watson at Sherlock and Lead Auditor for Trust Security and Renascence Labs.

Disclaimer

Smart contracts are an experimental technology with many known and unknown risks. Trust Security assumes no responsibility for any misbehavior, bugs or exploits affecting the audited code or any part of the deployment phase.

Furthermore, it is known to all parties that changes to the audited code, including fixes of issues highlighted in this report, may introduce new issues and require further auditing.

Methodology

In general, the primary methodology used is manual auditing. The entire in-scope code has been deeply looked at and considered from different adversarial perspectives. Any additional dependencies on external code have also been reviewed.

Findings

Low severity findings

TRST-L-1 apxETH collateral should be deployed with higher DELAY_UNTIL_DEFAULT parameter

- **Category:** Configuration issues
- **Source:** deploy_apxeth.ts
- **Status:** Fixed

Description

The internal discussion by the Reserve team on the apxETH PR suggests that **DELAY_UNTIL_DEFAULT** should be equal to 72 hours instead of 24 hours.

Recommended mitigation

Consider changing **DELAY_UNTIL_DEFAULT** from 24 hours to 72 hours.

Team response

Fixed by changing **DELAY_UNTIL_DEFAULT** to 72 hours**.** The higher **DELAY_UNTIL_DEFAULT** reflects the lower liquidity of apxETH compared to other liquid staking tokens.

Mitigation review

Fixed as recommended.

TRST-L-2 apxETH collateral should be deployed with lower DEFAULT_THRESHOLD parameter

- **Category:** Configuration issues
- **Source:** deploy_apxeth.ts
- **Status:** Fixed

Description

The **DEFAULT_THRESHOLD** for the apxETH collateral is currently set to 5%. According to the client, this is to allow for withdrawal fees to be increased for pxETH / ETH.

However, increased withdrawal fees for apxETH / pxETH can already cause a hard default due to the decrease in *_underlyingRefPerTok()*, and so to allow for withdrawal fees in pxETH and not in apxETH is inconsistent.

Another consideration is that when the pxETH / ETH withdrawal fee is increased, this decreases the effective peg price, meanwhile the Reserve contracts still assume the peg price is one. It is therefore not recommended to allow for a big increase in withdrawal fees since the Reserve contracts won't reflect it.

Recommended mitigation

Consider changing **DEFAULT_THRESHOLD** to a lower value. Historically, a 2% threshold has been sufficient to not mark the collateral as defaulted.

Team response

Fixed. The default threshold is now 2% plus the oracle error. Overall, it is 3%.

Mitigation review

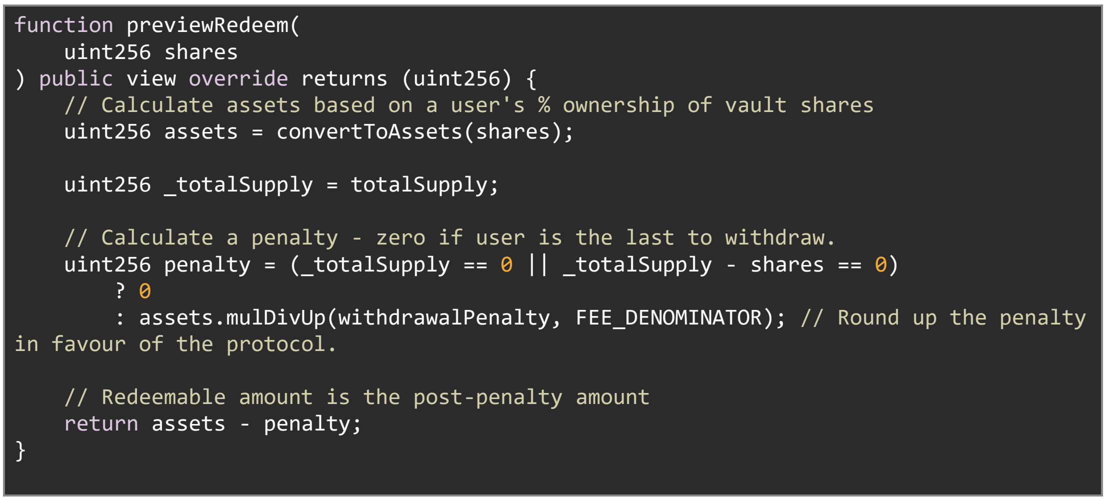
Fixed as recommended.

TRST-L-3 apxETH withdrawal penalty can be enabled which decreases ref per tok rate and disables the collateral

- **Category:** Configuration issues
- **Source:** ApxEthCollateral.sol
- **Status:** Acknowledged

Description

ApxEthCollateral.underlyingRefPerTok() calls *apxETH.assetsPerShare()* which in the downstream calls apxETH.*previewRedeem()*. This function takes into consideration a withdrawal penalty.



```
function previewRedeem(
    uint256 shares
) public view override returns (uint256) {
    // Calculate assets based on a user's % ownership of vault shares
    uint256 assets = convertToAssets(shares);

    uint256 _totalSupply = totalSupply;

    // Calculate a penalty - zero if user is the last to withdraw.
    uint256 penalty = (_totalSupply == 0 || _totalSupply - shares == 0)
        ? 0
        : assets.mulDivUp(withdrawalPenalty, FEE_DENOMINATOR); // Round up the penalty in favour of the protocol.

    // Redeemable amount is the post-penalty amount
    return assets - penalty;
}
```

Currently the fee is zero but if it is enabled to any reasonable value, the revenue hiding of **1e-4** is not sufficient and the collateral defaults immediately. Its maximum value, enforced by the apxETH contract, is 5%.

Recommended mitigation

The issue can be addressed in different ways:

1. Do nothing and assume the fee won't be enabled or accept the collateral will be disabled in such a case.
2. Increase revenue hiding to a value less than 5% by assuming a lower maximum fee than enforced by the contract (setting revenue hiding to the full 5% probably makes the plugin unusable).

3. Calculate an exchange rate that is not affected by the withdrawal fee. *convertToAssets(1e18)* is correct for this purpose. However, in this case the ref per tok rate won't reflect the actual rate at which apxETH can be exchanged for pxETH.

Team response

Acknowledged. The collateral plugin will become disabled if the withdrawal penalty is increased.

Mitigation review

The finding has been acknowledged. The collateral will become **DISABLED** if the withdrawal penalty is enabled.

TRST-L-4 sUSDe collateral should be deployed with lower DEFAULT_THRESHOLD parameter

- **Category:** Configuration issues
- **Source:** deploy_USDe.ts
- **Status:** Fixed

Description

The default threshold in sUSDe is set to 5%. According to the client, this is intended and different versions of the sUSDe collateral plugin will be deployed with different default thresholds. This is due to internal discussions about the mechanics of the USDe token.

During the audit of the 4.0.0 release however, it has been recognized that large default thresholds make the RToken vulnerable to arbitrage in case of a de-peg, and 5% is an unusually large default threshold.

Recommended mitigation

Consider lowering the default threshold and to make it similar to other collateral plugins.

Team response

Fixed. The default threshold is now 1% plus the oracle error. Overall, it is 1.5%.

Mitigation review

Fixed as recommended.

Additional recommendations

Comment for sUSDe DELAY_UNTIL_DEFAULT is incorrect

In the deployment script for the sUSDe collateral it is stated that **DELAY_UNTIL_DEFAULT** is equal to 24 hours. However, **DELAY_UNTIL_DEFAULT** is defined as 72 hours which according to the client is correct. Therefore, the recommendation is to adjust the comment to 72 hours.

apxETH verification script verifies wrong address

The verification script for the apxETH collateral uses the address for the **rETH** collateral. This needs to be changed to **apxETH**.

Systemic risks

RToken backing can be affected by legal restrictions

RTokens can technically be backed by any ERC20 token. However, technical compatibility does not ensure legal compliance. RToken Governance as well as users interacting with RTokens must ensure that they comply with regulations. Many tokens have blacklist functionality and non-compliant usage of the token can lead to a freezing and thereby a loss of funds.