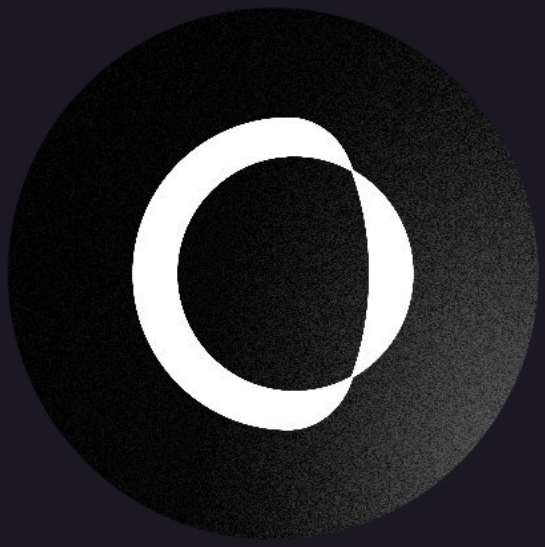
SHERLOCK

Security Review For Usual



Collaborative Audit Prepared For: **Usual**
Lead Security Expert(s): **0x73696d616f**
eeyore
Date Audited: **November 11 - November 14, 2025**

Introduction

bASSET0 & rt-ASSET0 tokens are the evolution of the current USD0++ model. This security review focuses on improving the user experience of redeeming for rt-USD0 holders and the upgrade's operational clean-up for technical debt reduction.

Scope

Repository: usual-dao/core-protocol

Audited Commit: 944105857d4af203c16137bca58f130472fd195e

Final Commit: b72162e963cd0a5bb86c0c93d1f2a7bfb63ad6dd

Files:

- src/constants.sol
- src/oracles/AbstractOracle.sol
- src/oracles/ClassicalOracle.sol
- src/registry/RegistryAccess.sol
- src/registry/RegistryContract.sol
- src/TokenMapping.sol
- src/token/RTUsd0.sol
- src/token/Usd0PP.sol
- src/token/Usd0.sol
- src/utils/CheckAccessControl.sol

Final Commit Hash

b72162e963cd0a5bb86c0c93d1f2a7bfb63ad6dd

Findings

Each issue has an assigned severity:

- High issues are directly exploitable security vulnerabilities that need to be fixed.
- Medium issues are security vulnerabilities that may not be directly exploitable or may require certain conditions in order to be exploited. All major issues should be addressed.
- Low/Info issues are non-exploitable, informational findings that do not pose a security risk or impact the system's integrity. These issues are typically cosmetic or

related to compliance requirements, and are not considered a priority for remediation.

Issues Found

High	Medium	Low/Info
0	0	3

Issues Not Fixed and Not Acknowledged

High	Medium	Low/Info
0	0	0

Issue L-1: `Usd0PP.sol::_deconstruct()` is always called with `msg.sender` despite having receiver arguments [RESOLVED]

Source: https://github.com/sherlock-audit/2025-11-usual-usd0-upgrade-nov-11th/issues/14

Summary

`Usd0PP.sol::_deconstruct()` is currently always called with `msg.sender` for both receiver arguments, so separate functions to mint with flexible receiver are missing.

Vulnerability Detail

The function is defined as `function _deconstruct(uint256 amountUsd0, address bAssetRecipient, address rAssetRecipient)`, but in the code only `Usd0PP.sol::mint()` and `Usd0PP.sol::mintWithPermit()` are exposed, and they don't allow to specify different recipients, so the functionality is not fully implemented.

Impact

Missing functionality.

Code Snippet

https://github.com/sherlock-audit/2025-11-usual-usd0-upgrade-nov-11th/blob/main/core-protocol/src/token/Usd0PP.sol#L226-L239

Tool Used

Manual Review

Recommendation

Add separate mint functions that allow receivers other than `msg.sender`.

Issue L-2: Usd0PP retains deprecated airdrop exit & Curve PAR logic after cleanup [RESOLVED]

Source: https://github.com/sherlock-audit/2025-11-usual-usd0-upgrade-nov-11th/issues/16

Summary

`Usd0PP` still exposes imports, roles, errors, storage slots, and public methods that belong to the removed airdrop exit path and Curve PAR mechanism.

Vulnerability Detail

The ongoing “Exiting deprecated airdrop logic” and “Pool balance Curve PAR removal” efforts left multiple externally callable functions and state variables intact:

- Imports and role IDs such as `IAirdropDistribution`, `PEG_MAINTAINER_ROLE`, `EARLY_BOND_UNLOCK_ROLE`, the Curve pool constants, and PAR-specific errors remain declared even though the features they supported were removed.
- Storage slots for `bondEarlyUnlockStart/End`, `bondEarlyUnlockAllowedAmount`, and `bondEarlyUnlockDisabled` are still live, along with the corresponding management functions (`setupEarlyUnlockPeriod`, `allocateEarlyUnlockBalance`, `setBondEarlyUnlockDisabled`, and the getter trio). These functions are callable, emit events, and enforce access roles that are supposed to be deprecated.
- PAR-specific artifacts like `ICurvePool` and the `PARNot*` error types persist, signaling incomplete removal of the Curve balancing code path.

Keeping these contracts accessible means governance must still maintain obsolete roles and ACL entries, and upgrade auditors cannot tell whether the migration was completed. Attackers (or misconfigured automation) can interact with these paths and find divergence between documentation and behavior.

Impact

Deprecated functionality was not removed correctly.

Code Snippet

https://github.com/sherlock-audit/2025-11-usual-usd0-upgrade-nov-11th/blob/main/core-protocol/src/token/Usd0PP.sol#L256 https://github.com/sherlock-audit/2025-11-usual-usd0-upgrade-nov-11th/blob/main/core-protocol/src/token/Usd0PP.sol#L403

Tool Used

Manual Review

Recommendation

Remove the legacy imports, role IDs, errors, and public methods tied to the deprecated airdrop exit and Curve PAR systems.

Storage layout must be preserved, rename the slots to explicit `unused*` fields and eliminate the callable logic plus related access roles.

Issue L-3: RTUsd0 minor cleanup gaps [RESOLVED]

Source: https://github.com/sherlock-audit/2025-11-usual-usd0-upgrade-nov-11th/issues/17

Summary

`RTUsd0` ships with avoidable code-quality regressions: an unused upgradeable reentrancy guard import, a redundant cast to `IUsd0`, and a `public` initializer that could be marked `external`.

Vulnerability Detail

- `ReentrancyGuardUpgradeable` is imported but never used, bloating bytecode and signaling incomplete refactors.
- `_update()` copies `$.usd0` into `IUsd0 usd0 = $.usd0;`, even though the storage slot already holds the correct type, so the reassignment is a no-op.
- `initialize()` remains `public`, which is broader than necessary for an initializer callable only once via the proxy admin.

Impact

Minor issues accumulate technical debt.

Code Snippet

https://github.com/sherlock-audit/2025-11-usual-usd0-upgrade-nov-11th/blob/main/core-protocol/src/token/RTUsd0.sol#L4-L6 https://github.com/sherlock-audit/2025-11-usual-usd0-upgrade-nov-11th/blob/main/core-protocol/src/token/RTUsd0.sol#L110 https://github.com/sherlock-audit/2025-11-usual-usd0-upgrade-nov-11th/blob/main/core-protocol/src/token/RTUsd0.sol#L153

Tool Used

Manual Review

Recommendation

Drop the unused `ReentrancyGuardUpgradeable` import, remove the redundant `IUsd0 usd0 = $.usd0;` assignment (use `$.usd0` directly), and tighten `initialize()` visibility to `external`.

Disclaimers

Sherlock does not provide guarantees nor warranties relating to the security of the project.

Usage of all smart contract software is at the respective users' sole risk and is the users' responsibility.