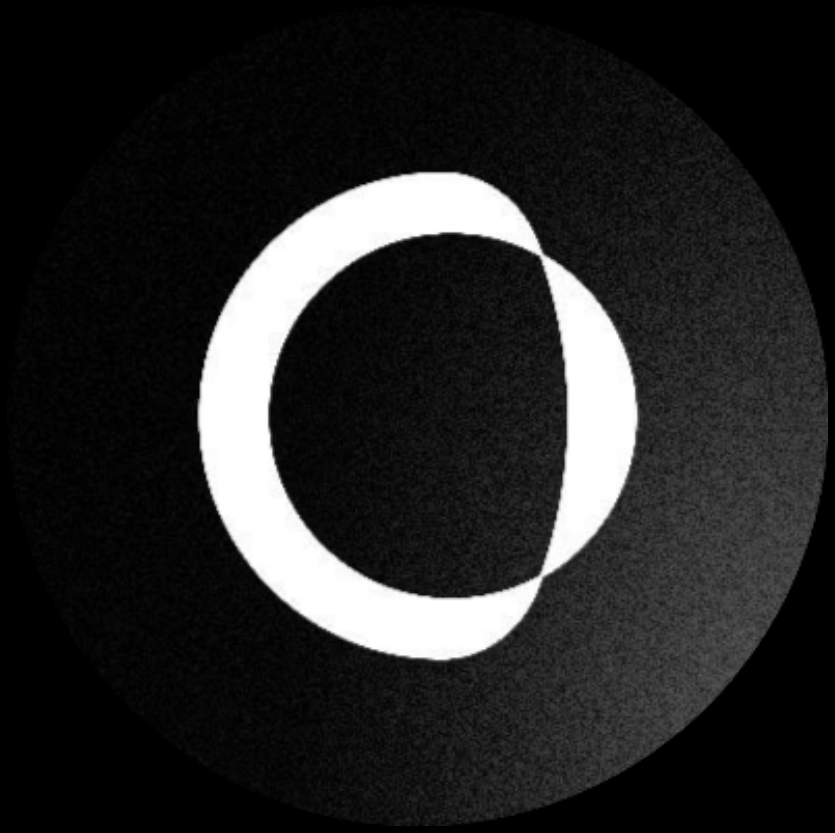
Usual
Security Review



NOVEMBER, 2024

www.chaindefenders.xyz
https://x.com/ChDefendersEth

Lead Auditors

PeterSR

0x539.eth

Table of Contents

- Table of Contents
- Protocol Summary
- Disclaimer
- Risk Classification
- Audit Details
  - Scope
  - Roles
- Executive Summary
  - Issues found
- Findings
  - High

Protocol Summary

Usual is a decentralized stablecoin issuer launched 3 months ago, now ranked among the top 15 with over $350M in TVL and 20k holders. The V1 release, which is the focus of this audit, includes the introduction of $USUAL, the governance token for the protocol, along with its allocation, staking and distribution logic, and the related airdrop contracts.

Disclaimer

The ChainDefenders team makes all effort to find as many vulnerabilities in the code in the given time period, but holds no responsibilities for the findings provided in this document. A security audit by the team is not an endorsement of the underlying business or product. The audit was time-boxed and the review of the code was solely on the security aspects of the Solidity implementation of the contracts.

Risk Classification

Likelihood/Impact	High	Medium	Low
High	H	H/M	M
Medium	H/M	M	M/L
Low	M	M/L	L

Audit Details

Scope

Id	Files in scope
1	AirdropDistribution.sol
2	AirdropTaxCollector.sol
3	constants.sol
4	DaoCollateral.sol
5	DistributionModule.sol
6	errors.sol
7	RewardAccrualBase.sol
8	Usd0.sol
9	Usd0PP.sol
10	Usual.sol
11	UsualS.sol
12	UsualSP.sol
13	CheckAccessControl.sol
14	NoncesUpgradeable.sol
15	normalize.sol
16	UsualX.sol
17	YieldBearingVault.sol

Roles

Id	Roles
1	User
2	Owner

Executive Summary

Issues found

Severity	Count	Description
High	1	Critical vulnerabilities
Medium	0	Significant risks
Low	0	Minor issues with low impact
Informational	0	Best practices or suggestions
Gas	0	Optimization opportunities

Findings

High

High 01 Reward is not updated during `removeOriginalAllocation`

Summary

In the `UsualSP` contract, rewards are based on the user's staked amount. Users can stake or unstake tokens, with their effective balance calculated as `amountStakedByUser + originalAllocation - originalClaimed`. The `originalAllocation` can be updated by users with the `USUALSP_OPERATOR_ROLE` via the `allocate` function, and `originalClaimed` tracks the claimed portion of the allocation. However, when `removeOriginalAllocation` is called, the rewards for users affected by the removal are not updated accordingly.

Root Cause

The issue stems from the absence of a reward update call in `removeOriginalAllocation`. Specifically, `_updateReward(recipients[i]);` is not called, resulting in outdated rewards for users whose allocations are removed.

External Pre-conditions

The root cause lies in the missing `_updateReward` call in `removeOriginalAllocation`, which prevents rewards from being updated when allocations are removed.

Impact

Users whose allocations are removed experience discrepancies in reward calculations. Their accumulated rewards do not reflect the allocation removal, potentially leading to financial loss and an unfair reward distribution.

PoC

Consider the following example:

- User's balance: 100:
  - 50 tokens staked.
  - 60 tokens as original allocation.
  - 10 tokens already claimed.
- Current `rewardPerToken`: `10 * 1e24`.
- User's `lastRewardPerTokenUsed`: `3 * 1e24`.

If `_updateReward` is not called during `removeOriginalAllocation`, the user's reward balance will exclude the additional reward from the 7 * 1e24 multiplier on their 100-token balance.

1. Expected Reward Calculation (with `_updateReward`):
   - Reward Adjustment:

     [(rewardPerToken - lastRewardPerTokenUsed) ×balance]
     Plugging in values:
     [(10 ×1e24 - 3 ×1e24) ×100 / 1e24 = 700]
     With `_updateReward`, the reward increases by `700`.
2. Current Behavior (without `_updateReward`):
   - The adjustment of `700` is not applied, leaving the reward balance inaccurate and resulting in user loss.
3. User Re-stakes in the Same Block:
   - User's new balance: 50 tokens.
   - Reward calculation: Based only on 50 tokens.

     [50 ×7 / 1e24 = 350]
     Instead of receiving the full `700`, the user gets only `350`.

Mitigation

Add a call to `_updateReward` in `removeOriginalAllocation`:

```
function removeOriginalAllocation(address[] calldata recipients)
    external {
        if (recipients.length == 0) {
            revert InvalidInputArraysLength();
        }

        UsualSPStorageV0 storage $ = _usualSPStorageV0();
        $.registryAccess.onlyMatchingRole(USUALSP_OPERATOR_ROLE);

        for (uint256 i; i < recipients.length;) {
+           _updateReward(recipients[i]);
            $.originalAllocation[recipients[i]] = 0;
            $.originalClaimed[recipients[i]] = 0;

            emit RemovedOriginalAllocation(recipients[i]);
            unchecked {
                ++i;
            }
        }
    }
```