OpenZeppelin | security

Origin WOETH and Vault Update

ØRIGIN

April 25, 2025

Table of Contents

Summary

Type	DeFi/Stablecoin
Timeline	From 2025-03-31 To 2025-04-03
Languages	Solidity

Total Issues	5 (4 resolved)
Critical Severity Issues	0 (0 resolved)
High Severity Issues	0 (0 resolved)
Medium Severity Issues	0 (0 resolved)
Low Severity Issues	1 (0 resolved)
Notes & Additional Information	1 (1 resolved)
Client Reported Issues	3 (3 resolved)

Scope

We audited pull request #2452 at commit f91a6ed and pull request #2453 at commit 8b237c0 of the OriginProtocol/origin-dollar repository.

In scope were the following files:

```
PR #2452
contracts/contracts/
├── interfaces
│   └── IVault.sol
└── vault
    ├── VaultAdmin.sol
    ├── VaultCore.sol
    ├── VaultInitializer.sol
    └──  VaultStorage.sol

PR #2453
contracts/contracts/
└── token
    └── WOETH.sol
```

System Overview

The update to the `WOETH` and Vault contracts prevents rapid inflation of the `WOETH` share-to-asset ratio and the rebasing rate of the OETH token. This allows `WOETH` to be used as a base asset on lending platforms, as many lending platforms are incompatible with `ERC4626` assets which can be rapidly inflated.

The Origin team had previously tried different approaches to prevent `WOETH` inflation. However, even when direct donations to the `WOETH` contract were mitigated, initiating a large rebase of the underlying `OETH` tokens also caused the `WOETH` token to inflate. Hence, the update alters the `OETH` contracts so that large rebases do not occur. Additionally, `WOETH` was updated to rely on internal accounting rather than directly querying the `balanceOf` the contract which prevents direct donations from inflating the share-to-asset ratio.

Within the `VaultCore` contracts, two measures were introduced to prevent rapid inflation and rebasing:

1. A maximum rebase rate. This limits the rebase amount per second designed to allow yield that are triple digit APR's in extreme conditions but still prevent rapid inflation.
2. A maximum amount per rebase. Even with the rebase rate limitation, a large period without any rebasing could be followed by a large rebase in a single transaction. To prevent this, a rebase is not permitted to exceed a hardcoded amount in a single rebase. Since the vault can only rebase once per block, this also prevents a large rebase in a single block.

The OETH dripper logic which smooths out yield distributions is now part of `VaultCore`. A target rate of yield is set based on previous yield. If the yield rate can sustain a larger yield rate, this target rate is increased. Conversely, if the target rate cannot be sustained the yield is decreased. This happens in addition to the rebase caps. The yield is first calculated through the dripper logic and then limited by the maximum rebase checks.

Security Model and Privileged Roles

The WOETH share-to-asset ratio should not be rapidly inflatable. However, protocols that integrate this asset should be aware that the share-to-asset ratio is expected to gradually inflate due to collecting rebasing yield from the underlying asset. It should also be noted that the governor can remove any asset aside from OETH from the WOETH contract.

Low Severity

L-01 Slight Overcharging Of Fees When Supply Cap Is Reached

In the `changeSupply` function, if the new `totalSupply` exceeds `MAX_SUPPLY`, any amount beyond `MAX_SUPPLY` is discarded, and the supply is limited to the `MAX_SUPPLY` value. If the fee mint were to bring `totalSupply` close enough to the supply cap, then `changeSupply` would exclude any yield above the maximum supply. The fee is still collected for the full yield amount despite it being limited by the supply cap.

This scenario is unlikely and the yield loss is minimal, so consider simply acknowledging this issue.

Update: *Acknowledged, not resolved. The Origin team stated:*

> *Due to low impact, we acknowledged this issue.*

Notes & Additional Information

N-01 Function Visibility Overly Permissive

Within `WOETH.sol`, there are various functions with unnecessarily permissive visibility:

- The `name` function in `WOETH.sol` with `public` visibility could be limited to `external`.
- The `symbol` function in `WOETH.sol` with `public` visibility could be limited to `external`.
- The `totalAssets` function in `WOETH.sol` with `public` visibility could be limited to `external`.

To better convey the intended use of functions, consider changing a function's visibility to be only as permissive as required.

Update: *Resolved, not an issue. The Origin team stated:*

> *The WOETH contract inherits from 4626 and that one inherits from ERC20 (https://github.com/OpenZeppelin/openzeppelin-contracts/blob/v4.4.2/contracts/token/ERC20/ERC20.sol). The ERC20 has `name` and `symbol` set to `public` (https://github.com/OpenZeppelin/openzeppelin-contracts/blob/v4.4.2/contracts/token/ERC20/ERC20.sol#L62-L72) and solidity compiler won't allow for overriding of function visibility. Similarly, 4626 has `totalAssets's` visibility set to `public` and can't be overridden.*

Client Reported

CR-01 Rate Should be Capped Using `<=` Rather Than `<`

It is more accurate to cap `newPerSecond` to be less than or equal to `MAX_REABSE_PER_SECOND` rather than strictly less than. To address this, the `<` inequality was replaced with `<=`.

Update: *Resolved in **pull request 2452** at **commit 7feda56**. The Origin team communicated this update during the audit period and this change was validated to be correctly implemented.*

CR-02 `totalAssets` Can Round In Favor Of The User

An attacker can mint discounted shares in an empty or low capacity vault due to the rounding behaviour of the `totalAssets` function. `totalAssets` always rounds down. It is used to calculate the number of shares on a deposit. A smaller assets value benefits a deposit, by giving the depositor more shares than they should receive. As the size of the deposit increases, the error is multiplied.

This allows the following attack sequence: 1. a tiny deposit to position `totalSupply` in a place that maximizes the rounding error in `totalAssets` 2. a very large deposit to exploit the rounding error, getting an unfair number of shares. (sized to end on `totalSupply` with minimal rounding error) 3. withdraw and profit 4. repeat

This attack can be looped, and each loop is more effective than the previous because the total supply is smaller after each attack, and the attacker has more funds after each attack. Both of these improve the percentage stolen on each loop.

This attack does require a large ratio between the `totalSupply` and the attack amount, and may be impractical once there's enough `totalSupply`.

Update: *Resolved in* ***pull request #2453*** *at* ***commit ebd7d3e****.*

CR-03 Initial Mint Can Be Incorrect

When `totalSupply` is zero, the wrapped token mints at a 1:1 ratio. This is correct only if no rebases have happened between initialization and the first mint. Otherwise the mint is at a privileged rate, and leaves the contract permanently insolvent, with more liabilities than assets.

Update: *Resolved in* ***pull request #2453*** *at* ***commit ebd7d3e****.*

Conclusion

This audit reviewed pull requests that further implemented security measures against token inflation attacks and to allow WOETH to be used as a base asset on lending platforms. The audit confirmed that the issues that were previously found in a prior audit are not present in the latest implementation. Communication with the Origin Protocol team was excellent throughout the audit period, with all questions being answered promptly and thoroughly.