Origin OUSD Audit

OPENZEPPELIN SECURITY | DECEMBER 18, 2024 Security Audits Solidity

Table of Contents

- Table of Contents
- Summary
- Scope
- System Overview
  - Yield Delegation
- Security Model and Trust Assumptions
  - Privileged Roles
- Low Severity
  - Users Lack Control Over Yield Delegation

- Notes & Additional Information
  - Redundant Code
  - Incomplete Docstrings
  - Missing Named Parameters in Mappings
  - Multiple Optimizable Storage Operations
  - Typographical Errors
- Conclusion

Summary

Type
DeFi
Timeline
From 2024-11-26
To 2024-12-02
Languages
Solidity

Total Issues
8 (4 resolved)
Critical Severity Issues
0 (0 resolved)
High Severity Issues
0 (0 resolved)
Medium Severity Issues

Notes & Additional Information
5 (3 resolved)

Scope

We audited the OriginProtocol/origin-dollar repository at commit 4495130.

In scope was the following file:

```
contracts
└── contracts
    └── token
        └── OUSD.sol
```

System Overview

This audit focuses on reviewing an upgrade to the existing `OUSD` contract, which implements a rebasing ERC-20 token. The primary objective of the upgrade is to enable yield delegation, allowing an account to seamlessly transfer all earned yield to another account. Additionally, the upgrade addresses minor rounding issues in the contract's internal accounting mechanism.

Previously, there were only two account types in the system: rebasing and non-rebasing.

- Rebasing accounts earn yield, causing their balances to increase over time. The majority of accounts in the system fall under this type.
- Non-rebasing accounts do not earn yield. These are primarily used by contracts that lack support for non-transfer balance changes or yield distribution.

The current upgrade introduces two new account types to enhance functionality:

- Yield delegation source accounts: these accounts do not earn yield directly but instead transfer their yield to a designated target account.
- Yield delegation target accounts: these accounts earn yield both from their own balance and from balances delegated to them by source accounts.

It is important to note that yield delegation operates as a 1:1 relationship. For instance, if Account A delegates yield to Account B, then A cannot delegate yield to any other account, and B cannot receive yield from any other source. Additionally, an account cannot simultaneously function as both a yield delegator and a receiver.

Security Model and Trust Assumptions

The upgrade reviewed in this audit introduces additional points for consideration:

- **Governor-controlled yield delegation**: Yield delegation is managed exclusively by the Governor. Consequently, accounts cannot voluntarily opt in or out of rebasing, nor can they choose who to delegate their yield to or receive yield from. This makes it critical for the Governor to act in good faith and honor users' intentions. Currently, the Governor is a Timelock contract, which enforces a delay between a proposal passing and its on-chain execution. Proposals can be created by holders of the xOGN token and must reach quorum to be approved.
- **User-favoring rounding mechanics**: Balances are rounded up in favor of users. Although the current rounding impact is negligible, it is essential to ensure that `rebasingCreditsPerToken` remains significantly higher than `1e18`, as per the current implementation. If this value were to decrease substantially, malicious users could exploit the system by inflating their balance through operations that repeatedly trigger rounding in their favor. For context, the current `rebasingCreditsPerToken` value is approximately `6.76e26`.

Privileged Roles

The system defines two roles with privileged access:

- **Vault role**: Authorized to mint and burn tokens from accounts and adjust the total supply of OUSD, effectively distributing yield to all rebasing accounts.

This audit assumes that the entities managing these roles always act as intended. Consequently, potential attacks or vulnerabilities involving misuse of these privileged roles were not considered within the scope of this audit.

Users Lack Control Over Yield Delegation

In the current implementation, the Governor role has full control over yield delegations, leading to several potential issues:

- The yield delegation process can be slow, as every delegation must pass quorum and is subject to the Timelock delay before it can be executed.
- Users have no direct control over their yield, relying entirely on the Governor to act in alignment with individual preferences. In extreme scenarios, the Governor could enforce undesirable delegations, such as requiring top OUSD holders to delegate their yield to a treasury under the Governor's control.
- Users may receive yield from accounts they did not consent to, potentially preventing them from receiving a higher yield from a desired source. Rectifying this would require submitting a governance proposal and waiting for the Timelock delay to elapse.

To address these concerns, consider granting users greater autonomy over their yield. For instance, the system could allow users to specify which accounts they wish to delegate to or receive yield from, as well as provide an option to opt in or out of yield delegation altogether.

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

Missing Docstrings

Within `OUSD.sol`, multiple instances of missing docstrings were identified:

- The `TotalSupplyUpdatedHighres` event
- The `AccountRebasingEnabled` event
- The `AccountRebasingDisabled` event
- The `Transfer` event
- The `Approval` event
- The `YieldDelegated` event
- The `YieldUndelegated` event
- The `totalSupply` state variable
- The `vaultAddress` state variable
- The `rebaseState` state variable
- The `yieldTo` state variable
- The `yieldFrom` state variable
- The `initialize` function
- The `symbol` function
- The `name` function
- The `decimals` function

clearly documented as well. When writing docstrings, consider following the Ethereum Natural Specification Format (NatSpec).

Update: Resolved in pull request #2319.

Floating Pragma

Pragma directives should be fixed to clearly identify the Solidity version with which the contracts will be compiled. However, the `OUSD.sol` file has the `solidity ^0.8.0` floating pragma directive.

To improve reliability and avoid unintended issues caused by differences between compiler versions, consider using a fixed pragma directive.

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

> *For now, we will keep using the Solidity version in our build configuration since that version works better with our current tooling.*

Notes & Additional Information

Redundant Code

- The `initialize` function cannot be executed since the contract was already initialized in previous versions. To avoid unnecessarily increasing code size, consider removing the function or commenting it out for future reference.
- In the `delegateYield` function, checking both the `yieldFrom` / `yieldTo` mappings and the `rebaseState` mapping is redundant, as the first check will only pass if the second passes, and vice versa. Consider removing the first check, as it involves more storage reads than the second one.
- In the `undelegateYield` function, there is no need to set the credit balance of the delegation source, as it was already set to their balance within `delegateYield`.

Consider removing these redundancies to enhance the clarity and efficiency of the codebase.

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

> *We require the `initialize` function for future token contract deployments that inherit `OUSD.sol`. Moreover, we intentionally set the credit balance in `undelegateYield` and check both mappings in `delegateYield` to prevent future changes from introducing an error in this critical part. Also, both of these functions are rarely called.*

Incomplete Docstrings

Consider thoroughly documenting all functions and events that are part of a contract's public API. When writing docstrings, consider following the Ethereum Natural Specification Format (NatSpec).

Update: Resolved in pull request #2321.

Missing Named Parameters in Mappings

Since Solidity 0.8.18, developers can utilize named parameters in mappings. This means mappings can take the form of `mapping(KeyType KeyName? => ValueType ValueName?)`. This updated syntax provides a more transparent representation of a mapping's purpose.

Within `OUSD.sol`, multiple instances of mappings without named parameters were identified:

- The `allowances` state variable
- The `creditBalances` state variable
- The `alternativeCreditsPerToken` state variable
- The `rebaseState` state variable
- The `yieldTo` state variable
- The `yieldFrom` state variable

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

> *We are using an older version of Solidity and cannot use named parameters in mappings at this time.*

Multiple Optimizable Storage Operations

Multiple optimizable storage reads and writes were identified in the `OUSD` contract:

- In the `_creditsPerToken` function, the `alternativeCreditsPerToken` mapping is accessed twice for the same key.
- In the `changeSupply` function, the `totalSupply`, `rebasingCredits_`, and `rebasingCreditsPerToken_` state variables are read multiple times.
- In the `undelegateYield` function, the `yieldTo` mapping is accessed twice for the same key.
- Within the `_adjustAccount` function, the `alternativeCreditsPerToken` for non-rebasing accounts is always set to `1e18`, even when the mapping already has that value.

To lower gas consumption, consider reducing unnecessary storage reads by caching these values in memory variables, and only write to storage when the value needs to be updated.

> - `_creditsPerToken` *has been optimized as suggested.*
> - `changeSupply` *is rarely called and, in this instance, we prefer code readability over gas optimization.*
> - `undelegateYield` *is rarely called and, in this instance, we prefer code readability over gas optimization.*
> - `_adjustAccount` *has been optimized as suggested.*

Typographical Errors

Throughout the OUSD contract, there are multiple instances of incorrect documentation and typographical errors:

- On line 411, "therefor" should be "therefore".
- On line 564, "their" should be "its".
- On line 427, "Address" should be "Balance".

Consider correcting these instances to improve the quality of the documentation.

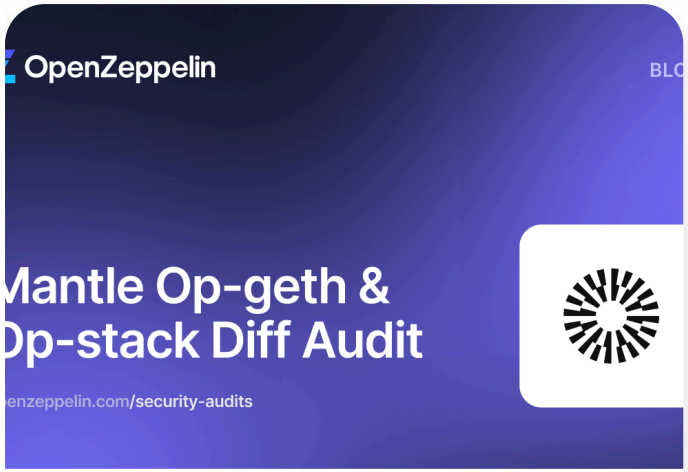
Update: *Resolved in pull request #2323.*

Conclusion

This audit focused on an upgrade to the OUSD contract, with the primary change being the ability to delegate yield from one account to another. We did not identify any major issues with the upgrade; it remains compatible with the previous version, and the contract should continue to function as intended following the implementation update. While some cases result in the protocol rounding balances in favor of users, the resulting gains are negligible and unlikely to pose any issues or be exploitable by malicious actors.

We provided several recommendations to enhance the overall quality of the codebase. The Origin team was cooperative and responsive to our questions throughout the audit process.

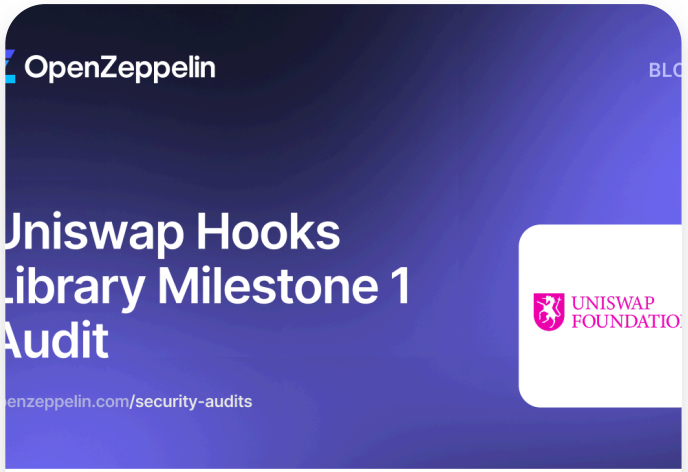
Related Posts



Mantle Op-geth & Op-stack Diff Audit

This audit is divided into three parts: A diff audit of the mantlenetworkio/mantle-v2 repo, a diff...

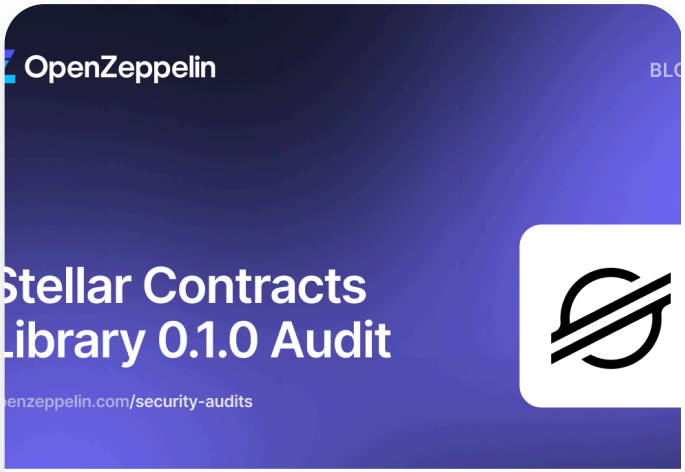
Security Audits L2's Go



Uniswap Hooks Library Milestone 1 Audit

We audited the OpenZeppelin/uniswap-hooks repository at commit f051c14. Smart contract audit.

Security Audits Solidity Uniswap



Stellar Contracts Library 0.1.0 Audit

We audited the OpenZeppelin/stellar-contracts repository at commit 01dbcb5. Smart Contract Audit.

Security Audits Rust

Terms of Service
Privacy Policy
Trademark Guidelines

Products

- Contracts Library
- Contracts Wizard
- Upgrade Plugins
- Defender Cloud Services

Services

- Smart Contract Security Audit
- Emergency Response
- ZKP Practice
- Ecosystem Stack

Resources

- Documentation
- Blog
- Forum
- Ethernaut CTF

Company

- About us
- Careers
- Brand Kit
- Trust