Euler usual

Security Review

Cantina Managed review by:

Xmxanuel, Lead Security Researcher
Om Parikh, Security Researcher

February 26, 2025

Contents

1 Introduction **2**
- 1.1 About Cantina . . . 2
- 1.2 Disclaimer . . . 2
- 1.3 Risk assessment . . . 2
  - 1.3.1 Severity Classification . . . 2

2 Security Review Summary **3**

3 Findings **4**
- 3.1 High Risk . . . 4
  - 3.1.1 `Euler Vault` breaks temporary USD0's 1:1 RWA Backing Rule and requires manual steps . . . 4
  - 3.1.2 Automatic debt token minting into the `treasury` can be exploited to `mint` new `USD0` if the protocol is undercollateralized . . . 5
- 3.2 Medium Risk . . . 6
  - 3.2.1 Missing integration tests for the `Euler` vaults . . . 6
  - 3.2.2 `oracle.getQuote` will be incorrect if `USD0` borrow positions are not liquidated in time with implications for `USD0.mint` . . . 6
- 3.3 Low Risk . . . 7
  - 3.3.1 `debt token` can be redeemed with `USD0` from `DAOCollateral` . . . 7
  - 3.3.2 User can front-run oracle switch to escape liquidation . . . 7
- 3.4 Informational . . . 8
  - 3.4.1 Predeployed oracle adapters `USD0USD` and `USD0PPUSD` are not based on secondary market prices . . . 8
  - 3.4.2 Replace zero addresses with mainnet addresses in deploy script . . . 8
  - 3.4.3 Borrowed `USD0` from Euler can be used to redeem RWAs during market turbulences . . . 8

1 Introduction

1.1 About Cantina

Cantina is a security services marketplace that connects top security researchers and solutions with clients. Learn more at cantina.xyz

1.2 Disclaimer

Cantina Managed provides a detailed evaluation of the security posture of the code at a particular moment based on the information available at the time of the review. While Cantina Managed endeavors to identify and disclose all potential security issues, it cannot guarantee that every vulnerability will be detected or that the code will be entirely secure against all possible attacks. The assessment is conducted based on the specific commit and version of the code provided. Any subsequent modifications to the code may introduce new vulnerabilities that were absent during the initial review. Therefore, any changes made to the code require a new security review to ensure that the code remains secure. Please be advised that the Cantina Managed security review is not a replacement for continuous security measures such as penetration testing, vulnerability scanning, and regular code reviews.

1.3 Risk assessment

Severity	Description
Critical	*Must* fix as soon as possible (if already deployed).
High	Leads to a loss of a significant portion (>10%) of assets in the protocol, or significant harm to a majority of users.
Medium	Global losses <10% or losses to only a subset of users, but still unacceptable.
Low	Losses will be annoying but bearable. Applies to things like griefing attacks that can be easily repaired or even gas inefficiencies.
Gas Optimization	Suggestions around gas saving practices.
Informational	Suggestions around best practices or readability.

1.3.1 Severity Classification

The severity of security issues found during the security review is categorized based on the above table. Critical findings have a high likelihood of being exploited and must be addressed immediately. High findings are almost certain to occur, easy to perform, or not easy but highly incentivized thus must be fixed as soon as possible.

Medium findings are conditionally possible or incentivized but are still relatively likely to occur and should be addressed. Low findings a rare combination of circumstances to exploit, or offer little to no incentive to exploit but are recommended to be addressed.

Lastly, some findings might represent objective improvements that should be addressed but do not impact the project's overall security (Gas and Informational findings).

2 Security Review Summary

Usual is a Stablecoin DeFi protocol that redistributes control and redefines value sharing. It empowers users by aligning their interests with the platform's success.

From Feb 4th to Feb 8th the Cantina team conducted a review of euler-usual on commit hash d836879d. The team identified a total of **9** issues:

Issues Found

Severity	**Count**	**Fixed**	**Acknowledged**
Critical Risk	0	0	0
High Risk	2	1	1
Medium Risk	2	2	0
Low Risk	2	1	1
Gas Optimizations	0	0	0
Informational	3	1	2
Total	**9**	**5**	**4**

3 Findings

3.1 High Risk

3.1.1 `Euler Vault` breaks temporary USD0's 1:1 RWA Backing Rule and requires manual steps

Severity: High Risk

Context: *(No context files were provided by the reviewer)*

Description: In the current Usual protocol, for every minted `1 USD0`, there exists a corresponding `$1 of RWA` value in the treasury in a healthy system state.

Usual wants to launch new Euler vaults that enables users to use `USD0++` as collateral to borrow `USD0`. Only `Usual Governance` can provide the required `USD0` liquidity to the vault. The Usual Governance can use their existing `USD0` holdings from fees, but as explained to the team, they also want to `mint` new `USD0` and therefore directly integrate it into their main protocol.

If Usual Governance decides to mint new `USD0` with a privileged call to provide liquidity to the Euler `USD0` vault, there technically exists no backing for the minted `USD0`.

Note: If no one borrows from the Euler vault, the unbacked `USD0` is not yet in circulation.

If a user wants to borrow the newly `unbacked` `USD0`, they need to provide `USD0++` as collateral. The `USD0++` bond represents a claim on locked `USD0` in the future and it's tradable on seconary markets. This means the `unbacked` `USD0` will now be in circulation after a `borrow` while the Euler vault holds `USD0++` as collateral. Technically, the underlying `USD0` of the `USD0++` collateral, which is `backed`, is locked and cannot be redeemed.

However, if there is too much `USD0` in existence compared to the dollar value of the `RWAs`, it becomes necessary to `burn` `USD0` to restore the RWA backing to a stable state. This should happen in the future either during `repayment` or `liquidation` after an Euler vault `borrow` transaction.

A `borrow` action in the Euler vault can have two outcomes: either the `debt` is repaid, or the `position` is liquidated. In the case of repayment, the initial `principal` amount of `USD0` should be `burned`. In the case of liquidation, Usual Governance would receive the `USD0++` collateral. In this scenario, Usual Governance plans to `unwrap` the `USD0++` and burn the borrowed principal amount of `USD0` as well.

Note: All these steps are planned to be performed manually by the Usual governance.

At a high level, as soon as unbacked `USD0` enters circulation, it would require to `burn` some other `USD0` to keep the packing. This other USD0 to be burned exists in the form of USD0++ collateral, which is locked and not in circulation, and will result in a USD0 burn in the future.

Technically, a special `debt token` will be minted into the Usual treasury as soon as `USD0` is borrowed from a vault. A new oracle is planned to assign a dollar value to the debt token, which should then be added as new collateral to the Usual protocol. However, the assigned debt token is merely a mechanism that allows the Usual protocol to handle this new system.

The debt token has no monetary value on secondary markets and could perhaps be seen as an abstract claim on Euler's `USD0++` collateral. However, a portion of the underlying `USD0` from `USD0++` tokens (between 0 and 83%) must be burned if the collateral is removed during a liquidation. Therefore, users cannot receive the corresponding Euler collateral `USD0++` value when redeeming USD0.

Impact: The new Euler vault mechanism is logically consistent but does create a temporary break in the 1:1 RWA backing model. The `eUSD0` vault temporarily violates the core value proposition of USD0 (1:1 RWA backing). If one of the unique propositions of USD0 is its permanent 1:1 RWA backing, users could lose confidence in it during this period, potentially triggering a bank run or a depeg event. However, such a temporary break of the rule can be acceptable if the community and token holders understand that the system will eventually "*correct*" it upon repayment or liquidation and aware of the additional risks and benefits.

Still, the new Euler vault involves a lot of risk. It is not guaranteed, that the system will "correct" the 1:1 RWA backing. If the Euler vault gets hacked not only the underlying `USD0++` would be stolen it would break the RWA backing permanently. In addition, the Euler governance plans to perform multiple crucial steps manually in this process. There exists no code that automatically burns `USD0` in the `repay` or liquidation case.

Therefore, the correctness of the individual steps is not guaranteed. In addition, the new model might enable other economics attacks on the protocol.

Recommendation: If the Usual Governance decides to launch this new feature, it should be carefully tested with integration tests and token economics simulations. The manual steps in the process should ideally be reduced, preferably with a specific contract that enforces a maximum ceiling for the newly minted `USD0`. The risk parameters need to be carefully monitored and the Governance needs to be able to react in market turbolences. Furthermore, we only recommend keeping the provided `USD0` liquidity to Euler at a level to not risk the overall stability of `USD0`.

Usual: Acknowledged on the technical & operational security being paramount to this feature. We have extensive safeguards to prevent any issues from arising, and will continue to implement them.

Our USL system is economically sound and has been vetted in several audits, including an economic audit on our whitepaper proving that the 1:1 RWA Invariant isnt broken on a protocol level, only during a transient state inbetween the repay/liquidation flow in 4 years

Cantina Managed: Acknowledged.

3.1.2 Automatic debt token minting into the `treasury` can be exploited to `mint` new `USD0` if the protocol is undercollateralized

Severity: High Risk

Context: HookTarget.sol#L71

Description: If a user borrows new `USD0` from the Euler vault, the `TargetHook` contract automatically mints new `debt tokens` into the Usual treasury. The `debt token` will be a new collateral type in the Usual system. Since there is no automatic `burning` of `debt tokens` in the `repay` case, the mechanism can be exploited to artificially increase the amount of `debt tokens` in the `treasury`.

An attacker only needs to `borrow` and `repay in the same transaction with for example a flash loan. This will artifically increase the` debt token`amount even when the`totalBorrows `= 0`. There is a high economic incentive to perform such an attack if the protocol is undercollateralized or if an RWA token drops in price. If the system is` undercollateralized`, it is not possible to mint new` USD0`tokens. The`mint`function checks whether the system is`undercollateralized`before minting new`USD0‘.

USD0 Mint Function

```
/// @dev Can only be called by an account with the USD0_MINT role.
function mint(address to, uint256 amount) public {
    // ...
    address[] memory rwas = $.tokenMapping.getAllUsd0Rwa();
    uint256 wadRwaBackingInUSD = 0;
    for (uint256 i = 0; i < rwas.length;) {
        address rwa = rwas[i];
        uint256 rwaPriceInUSD = uint256(oracle.getPrice(rwa));
        uint8 decimals = IERC20Metadata(rwa).decimals();

        wadRwaBackingInUSD +=
            Math.mulDiv(rwaPriceInUSD, IERC20(rwa).balanceOf(treasury), 10 ** decimals);

        unchecked {
            ++i;
        }
    }
    if (totalSupply() + amount > wadRwaBackingInUSD) {
        revert AmountExceedBacking();
    }
    _mint(to, amount);
}
```

However, with the above-mentioned attack, an attacker can create a stable state again by minting arbitrarily many debt tokens into the treasury. In the current Usual protocol, more than 90% of `USD0` is locked in the `USD0++` bond contract and is not available on the market. In case an RWA token depegs and the `CBR` (counter-bank run) mechanism in Usual is not activated, it would allow an arbitrage opportunity to mint new USD0 with the depegged RWA and redeem it for another RWA.

Recommendation: Evaluate if automatically burning the `debt token` in the Euler repay flow would make sense to prevent such an attack. Another option would be to not mint the `debt token` directly into the `UsualTreasury`. However, this would require another mechanism to move the correct `debt token` amount into the `UsualTreasury`.

Usual: Fixed in commit 652ad423 by switching from dUSD0 to eUSD0, which is the vaultshare of the USD0 vault, thereby not allowing automatic `debtToken` collateralization without consensus.

Cantina Managed: Fix verified.

3.2 Medium Risk

3.2.1 Missing integration tests for the `Euler` vaults

Severity: Medium Risk

Context: *(No context files were provided by the reviewer)*

Description: There are no `integration tests` verifying the basic functionality of the `Euler` Vaults at the time of this review. The `USD0` vault uses a custom TargetHook implementation and is not a standard Euler vault deployment. Furthermore, the `USD0` vault is planned to be integrated directly into the Usual protocol by allowing the `debt token` as a new collateral type.

Recommendation: All basic and edge case flows should be tested as RPC integration tests against the Usual mainnet deployment before launching the feature. All required manual steps should also be simulated and tested. During the security review the following basic integration tests were developed to ensure the correctness of the deployed Euler vault.

Usual: Fixed in commit daf58550.

Cantina Managed: Fixed.

3.2.2 `oracle.getQuote` will be incorrect if `USD0` borrow positions are not liquidated in time with implications for `USD0.mint`

Severity: Medium Risk

Context: *(No context files were provided by the reviewer)*

Description: The fix review included new version of the `vault` which uses the `vault shares` as collateral instead of the `debt token` in Usual. Using the Euler `vault shares` as collateral has the following implications:

In the current Euler vault setup, the LTV is set to `0.83`, and the LLTV is set to `0.9999`. If the debt of a position exceeds `0.9999` and is not liquidated, it impacts the value of `USD0` shares.

The collateral for a USD0 borrow transaction is `USD0++` and has a maximum value of `$1.00`. If the `debt` exceeds `$1.00`, it is unreasonable to assume the debt will be repaid, as the returned collateral would have a lower value.

However, `oracle.getQuote` or `USD0.convertToAssets` would still account for the outstanding debt as value for the shares.

For example:

```
USD0PP collateral value:  1.00
usd0 euler debt:          1.05
usd0 minted by usual:     0.83
```

The `value` of a single Euler USD0 share in the treasury would be incorrectly set at `1.05`.

This can cause issues when performing RWA backing check in the `USD0.mint` function. The RWA backing check can be successful in some cases (like the example above) when it should actually revert.

It would be possible to mint new USD0, when the protocol is depegged.

Simplified Case: Consider a scenario where there is only one other RWA token in the treasury, and `0.83` USD0 has been minted for the Euler vault. If the other RWA drops to `$0.78`, Usual can only receive a maximum of `1.00` USD0 back from Euler for the initially minted `0.83`.

```
usd0 supply:          1.83
rwa value:            0.78
rwa usd0 shares value: 1.05
total rwa value:      1.83
```

The RWA backing check in the `USD0.mint` function would pass. However, it should fail because the maximum value of the USD0 shares is `1.00`, not `1.05`. After liquidation, the difference between the collateral value and the `debt` of a borrower remains in the `debt` of the `borrower`. In the example above, this would be `0.05`. This leftover `debt` would continue to accumulate interest over time. Therefore, liquidation needs to occur when `LTV > 0.9999 && LTV < 1.00` to avoid the error.

Recommendation: For a completely correct RWA backing check the `USD0` minted by Usual for the Euler vault should be removed from the `totalSupply` together with the value of the `Euler` USD0 shares. The liquidation needs to happen between `LTV > 0.9999 && LTV < 1.00`. The time window for this will be only a few hours. The Usual governance needs to ensure that the liquidation will happen in time. A lower `LLTV`, such as 0.9900, could provide more time.

Usual: The window will be 17.5h to liquidate without taking on bad debt, which we will ensure with a liquidation bot.

Cantina Managed: Fixed.

3.3 Low Risk

3.3.1 `debt token` can be redeemed with `USD0` from `DAOCollateral`

Severity: Low Risk

Context: *(No context files were provided by the reviewer)*

Description: The `debt token` needs to get added as a new collateral type to Usual by calling `addUsd0Rwa` in the TokenMapping contract (see TokenMapping.sol#L85).

If no further modifications to the Usual protocol are implemented in the Usual protocol any user could redeem `debt tokens` by calling DaoCollateral.sol#L677. This `debt tokens` are indicated to be burned together with the `usd0` in the repay or liquidation case. However, this has no direct impact on the behavior of the system. The current debt token implementation is not upgraded and doesn't have a default admin role. There is no financial incentive for a user to perform a debt token redeem call.

Recommendation: Prevent users from redeeming `debt tokens` by introducing for example with an `allowlist`.

Usual: The collateral multisig is not giving an allowance to `DaoCollateral` to redeem `Vaultshares` (eUSD0) to prevent unexpected side effects, i.e. it cant be redeemed for.

Cantina Managed: Considering the above setting, the issue is fixed. It is still recommended to upgrade the `DAOCollateral` in the future to check the allowance in the `_burnStableTokenAndTransferCollateral` function and revert with a custom error (`CollateralNotRedeemable`) instead of an ERC20 allowance error.

3.3.2 User can front-run oracle switch to escape liquidation

Severity: Low Risk

Context: *(No context files were provided by the reviewer)*

Description: Currently `USD0PP` has oracle which has hard-coded price to `1 USD`, whenever protocol wants to make switch to oracle which is has dynamic rate and if reported rate is less than `1 USD`, It would allow users to escape liquidation by front-running oracle update. Front-running allows to settle position before LTV increases above liquidation threshold and hence allows user to repay less quantity for the same.

Proof of Concept:

- testSwitchOracle.

  Note: for ease of writing test, the price is being switched to 0.1 USD, but same can be justified with any other price too.

Recommendation: If possible, hardcoded rate oracles should be avoided from initial deployment else explore options to pausing system before creating oracle upgrade transaction.

Usual: Acknowledged. This will only be a relevant issue if we push the NAV oracle that is work in progress with Chainlink **only** when we are already depegged on our underlying RWA (US-Treasuries / reverse repos tokenized via USYC / M0 Token).

Cantina Managed: Acknowledged.

3.4 Informational

3.4.1 Predeployed oracle adapters `USD0USD` and `USD0PPUSD` are not based on secondary market prices

Severity: Informational

Context: Deploy.s.sol#L28-L29

Description: The Euler vault uses a predeployed `USD0` and `USD0PP` oracle. The Oracle currently has a fixed conversion rate for both (`USD0` and `USD0++`) at `$1`. The oracle does not reflect the secondary market price of `USD0` or `USD0++`. This means under market turbulence if the Euler vault parameters are not adjusted accordingly it will create arbitrage opportunities. For example the current LTV is 0.83 for borrowing USD0 with a USD0++ collateral. If the secondary market price of `USD0++` drops below `0.83` it would create an arbitrage opportunity.

Recommendation: Communicate the oracle parameters to the community and monitor the system carefully under market turbulences.

Usual: Acknowledged. `USD0++` currently holds a price of 0.9444 with a guaranteed minimum floorprice of 0.87 `USD0` per 1 `USD0++`. We are actively monitoring and will intervene if such an event still occurs despite several arbitrage mechanisms instantly raising the price to at least 0.87 via an active market.

Cantina Managed: Acknowledged.

3.4.2 Replace zero addresses with mainnet addresses in deploy script

Severity: Informational

Context: *(No context files were provided by the reviewer)*

Description/Recommendation: In deploy script:

```
// Usual addresses
address internal constant TREASURY = address(0); // TODO: set
address internal constant GOVERNOR = address(0); // TODO: set
```

this would lead to wrong deployment if not set correctly, make sure to set to correct mainnet addresses.

Usual: Fixed in commit 3b421684.

Cantina Managed: Fix verified.

3.4.3 Borrowed `USD0` from Euler can be used to redeem RWAs during market turbulences

Severity: Informational

Context: *(No context files were provided by the reviewer)*

Description: The borrowed `USD0` from Euler vaults can always be used to `redeem` RWA collateral. If the protocol is undercollateralized, it provides users with a way to redeem a portion of their `USD0` in the form of `USD0++` and exit the protocol. Currently, the circulating supply of `USD0` is low, as more than 93% of `USD0` is locked in `USD0++`.

Recommendation: This factor needs to be considered in economic evaluations, especially if the `LTV` is set to a high value. A high `LTV` would reduce the `RWA` in the treasury and, overall, increase the share of the debt token in the treasury. However, it would be not possible to redeem all the available RWAs from the treasury if the `LTV` is lower then `1`.

Usual: Acknowledged. The LTV is set at 83% at the maximum, in conjunction with the interest rate & maturity.

Cantina Managed: Acknowledged.