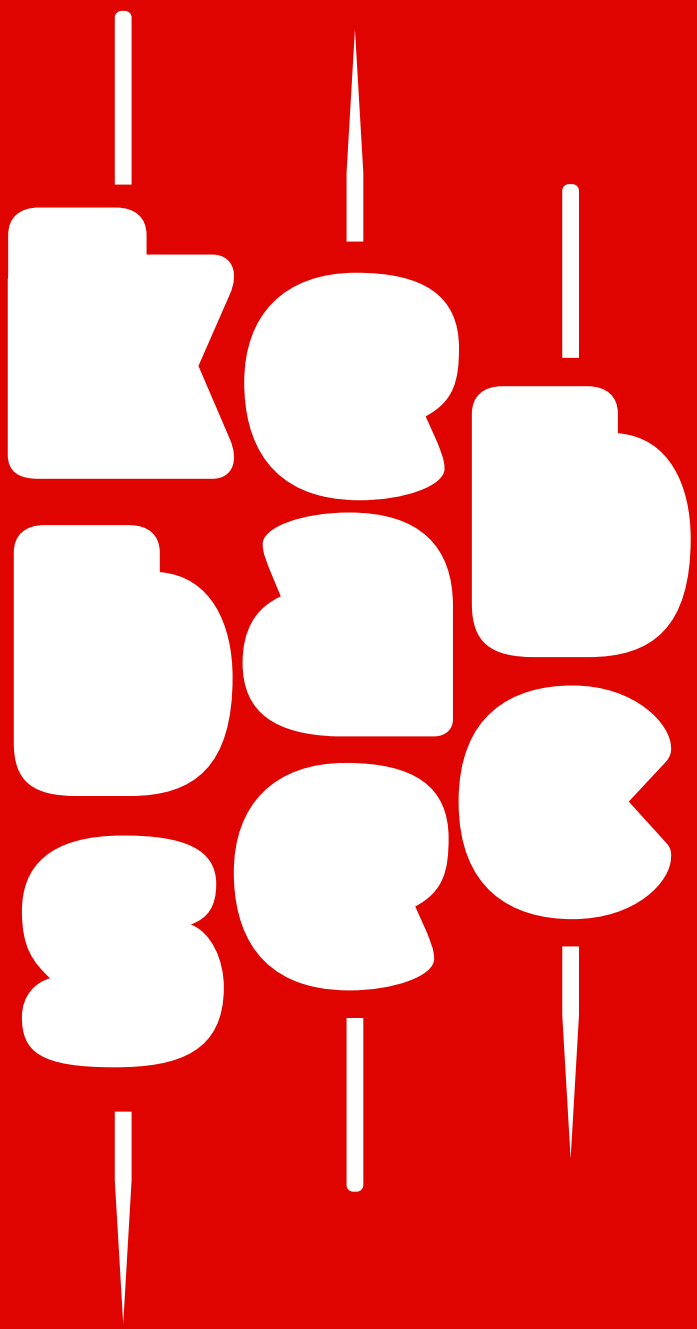

Contents

Introduction & Scope

This audit looks into the following contracts of `brrETH-upgradeable`: `BrrETH.sol` and `BrrETHRedeemHelper.sol` of the commit hash `2cc96565e2394ae9cb40db0f3d317ae1048285e0`.

This audit was conducted by `kebabsec` members `okkothejawa` and `ElCid`.

Note: This report does not provide any guarantee or warranty of security for the project.

Findings:

1. [LOW] Ownership structure is reduced to 1-step

Context: BrrETH.sol#L70-L71

Severity: Low

Description: The new access control modifier `onlyAdmin` used in the `BrrETH.sol` contract is utilizing the admin account registered in the `ERC1967Factory` as opposed to the `onlyOwner` of the immutable version of `brrETH` which uses a 2-step ownership scheme with a owner variable.

```
modifier onlyAdmin() {
    if (msg.sender != _ERC1967_FACTORY.adminOf(address(this)))
        revert ERC1967Factory.Unauthorized();

    _;
}
```

As the `ERC1967Factory`'s ownership logic is 1-step, this reduces the ownership structure of `brrETH` to 1-step ownership as well. The `ERC1967Factory`'s `changeAdmin` function is as follows:

```
/// @dev Sets the admin of the proxy.
/// The caller of this function must be the admin of the proxy on this
↪  factory.
function changeAdmin(address proxy, address admin) public {
    assembly {
        // Check if the caller is the admin of the proxy.
        if iszero(eq(sload(shl(96, proxy)), caller())) {
            mstore(0x00, _UNAUTHORIZED_ERROR_SELECTOR)
            revert(0x1c, 0x04)
        }
```

```
            // Store the admin for the proxy.
            sstore(shl(96, proxy), admin)
            // Emit the {AdminChanged} event.
            log3(0, 0, _ADMIN_CHANGED_EVENT_SIGNATURE, proxy, admin)
        }
    }
```

Recommendation: Re-implement 2-step ownership structure.

Client response: Acknowledged. As the `_ERC1967_FACTORY.adminOf(address(this))` can be assumed to be a multisig and it indeed is in the deploy script, the spirit of the issue is alleviated as it is harder to make ownership transfer mistakes with a multisig.

---

2. [LOW] `setCometRewards` should do approvals to router

Context: BrrETH.sol#L283-L292

Severity: Low

Description: The `setCometRewards` function is not approving the `router` to `cometRewards.rewardConfig(_COMET)` even though the reward token might be different in this new instance of `cometRewards`. This may result in `harvest` reverting as `router` may not be able to pull tokens. This can be resolved by calling `setRouter` with the current `router` address which mitigates the issue but this is not a proper solution.

```
    /**
     * @notice Set the Comet Rewards contract.
     * @param _cometRewards address Comet Rewards contract address.
     * @param shouldHarvest bool Whether to call `harvest` before
↪   setting `cometRewards`.
     */
    function setCometRewards(
        address _cometRewards,
        bool shouldHarvest
    ) external onlyAdmin {
        if (_cometRewards == address(0)) revert InvalidCometRewards();
        if (shouldHarvest) harvest();

        cometRewards = ICometRewards(_cometRewards);

        emit SetCometRewards(_cometRewards, shouldHarvest);
    }
```

Recommendation: Add an approval to the `router` for the `cometRewards.rewardConfig(_COMET)` in `setCometRewards`.

Client response: Acknowledged. A possible upgrade to `CometRewards` is expected which may bring multiple reward tokens, thus the client is waiting for that upgrade before making changes to the `setCometRewards` function as such an upgrade may require further changes.