Brett Audit Report

Version 1.0

Audited by:

HollaDieWaldfee

alexxander

June 14, 2024

Contents

1 Introduction

1.1 About Renascence

Renascence Labs was established by a team of experts including HollaDieWaldfee, MiloTruck, alexxander and bytes032.

Our founders have a distinguished history of achieving top honors in competitive audit contests, enhancing the security of leading protocols such as Reserve Protocol, Arbitrum, MaiaDAO, Chainlink, Dodo, Lens Protocol, Wenwin, PartyDAO, Lukso, Perennial Finance, Mute and Taurus.

We strive to deliver tailored solutions by thoroughly understanding each client's unique challenges and requirements. Our approach goes beyond addressing immediate security concerns; we are dedicated to fostering the enduring success and growth of our partners.

More of our work can be found here.

1.2 Disclaimer

This report reflects an analysis conducted within a defined scope and time frame, based on provided materials and documentation. It does not encompass all possible vulnerabilities and should not be considered exhaustive.

The review and accompanying report are presented on an 'as-is' and 'as-available' basis, without any express or implied warranties.

Furthermore, this report neither endorses any specific project or team nor assures the complete security of the project.

1.3 Risk Classification

	Impact: High	**Impact: Medium**	**Impact: Low**
Likelihood: High	High	High	Medium
Likelihood: Medium	High	Medium	Low
Likelihood: Low	Medium	Low	Low

1.3.1 Impact

- High - Funds are **directly** at risk, or a **severe** disruption of the protocol's core functionality
- Medium - Funds are **indirectly** at risk, or **some** disruption of the protocol's functionality
- Low - Funds are **not** at risk

1.3.2 Likelihood

- High - almost certain to happen, easy to perform, or not easy but highly incentivized
- Medium - only conditionally possible or incentivized, but still relatively likely
- Low - requires stars to align, or little-to-no incentive

2 Executive Summary

2.1 About Brett

Brett is an ERC20 token deployed on Base. The goal of the audit is to ensure that the live deployment of Brett is non-exploitable and can be used as a regular ERC20 token.

Non-exploitable includes non-exploitable by regular users but also non-exploitable by any privileged roles.

2.2 Overview

Project	Brett
Contract	0x532f27101965dd16442e59d40670faf5ebb142e4
Date	13 June 2024 - 14 June 2024

2.3 Issues Found

Severity	**Count**
High Risk	0
Medium Risk	0
Low Risk	0
Informational	3
Total Issues	**3**

3 Findings Summary

ID	Description	Status
I-1	`owner` role is revoked and privileged functions can never be called	Resolved
I-2	Custom logic in `BrettToken._transfer()` is cut short by setting fees to zero	Resolved
I-3	Transfer limits are disabled	Resolved

4 Findings

Informational

[I-1] `owner` role is revoked and privileged functions can never be called

Context:

- 0x532f27101965dd16442e59d40670faf5ebb142e4

Description: By inspecting the Brett token on Basescan, it can be observed that `owner` returns `address(0)`.

As a result, it will never be possible again to call any of the privileged functions: `addLiquidity()`, `enableTrading()`, `removeLimits()`, `setSwapEnabled()`, `setSwapTokensAtAmount()`, `setMaxWalletAndMaxTransaction()`, `setBuyFees()`, `setSellFees()`, `setMarketingWallet()`, `setDevelopmentWallet()`, `setLiquidityWallet()`, `excludeFromMaxTransaction()`, `bulkExcludeFromMaxTransaction()`, `excludeFromFees()`, `bulkExcludeFromFees()`, `withdrawStuckTokens()` and `airdrop()`.

The assessment of the deployed token can be performed at the current configuration since it can never change.

[I-2] Custom logic in `BrettToken._transfer()` is cut short by setting fees to zero

Context:

- 0x532f27101965dd16442e59d40670faf5ebb142e4

Description: By inspecting the Brett token on Basescan, it can be observed that `sellTotalFees` and `buyTotalFees` returns `0`.

As a result, `_tokensForLiquidity`, `_tokensForMarketing` and `_tokensForDevelopment` are not incremented and stay at their current value which is zeo.

```
if (takeFee) {
    // on sell
    if (_automatedMarketMakerPairs[to] && sellTotalFees > 0) {
        fees = amount.mul(sellTotalFees).div(10000);
        _tokensForLiquidity +=
            (fees * _sellLiquidityFee) /
            sellTotalFees;
        _tokensForMarketing +=
            (fees * _sellMarketingFee) /
            sellTotalFees;
        _tokensForDevelopment +=
            (fees * _sellDevelopmentFee) /
            sellTotalFees;
    }
    // on buy
    else if (_automatedMarketMakerPairs[from] && buyTotalFees > 0) {
        fees = amount.mul(buyTotalFees).div(10000);
        _tokensForLiquidity += (fees * _buyLiquidityFee) / buyTotalFees;
        _tokensForMarketing += (fees * _buyMarketingFee) / buyTotalFees;
        _tokensForDevelopment +=
            (fees * _buyDevelopmentFee) /
            buyTotalFees;
```

```
        }

        if (fees > 0) {
            super._transfer(from, address(this), fees);
        }

        amount -= fees;
    }
```

A downstream consequence is that in `_swapBack()`, the function is shortcut since `totalTokensToSwap = 0`.

```
    function _swapBack() internal {
        uint256 contractBalance = balanceOf(address(this));
        uint256 totalTokensToSwap = _tokensForLiquidity +
            _tokensForMarketing +
            _tokensForDevelopment;
        bool success;

        if (contractBalance == 0 || totalTokensToSwap == 0) {
            return;
        }
```

This behavior can be observed by any live transaction. For example, in this tenderly link: TX 0x5e43443094fd5621024d32ec39c5799073f33ca5c5d814fd9bd13618b0890fbf.

[I-3] Transfer limits are disabled

Context:

- 0x532f27101965dd16442e59d40670faf5ebb142e4

Description: By inspecting the Brett token on Basescan, it can be observed that `limited` returns `false`. `maxWallet` and `maxTransaction` return 1e10 which is bigger than the `totalSupply()` of slightly less than 1e10. `totalSupply()` can never grow, and so the balance and transaction limits are effectively disabled forever.

5 Centralization risks

5.1 Owner role is revoked and the token is fully permissionless

`BrettToken` inherits from OpenZeppelin's `Ownable` contract. However, the `owner` role has been revoked, and so the privileged functions cannot be called anymore.

As described in the findings section, by revoking the `owner` role and setting all fees to zero, the token behaves as a regular ERC20 token. All custom logic is cut short by the current contract configuration.