EUA: Security Audit Report

DogScan Security Team

DogScan

Nov 11th, 2025

Contents

DogScan Security Audit Report

Project	EUA Token
Chain	BNB Smart Chain (ID 56)
Smart Contract	`EUA`
Audit Date	2025-11-11
Report Version	1.0
Overall Risk	**Low**

1. Executive Summary

We performed a static review of `eua.sol`, the EUA token contract. It extends the OpenZeppelin ERC20 implementation with custom fee routing, a sell-side cooldown, a profit tax, and dedicated interactions with redemption contracts. Key behaviours include:

- Complex buy/sell fee structure (burn, LP reflow, special wallets)
- One-minute sell cooldown mechanism
- User cost-basis tracking with profit taxation
- Special hooks for redemption contracts such as `recycle`

Our review focused on fee accounting, cooldown logic, cost basis management, privilege control, and interactions with the Uniswap router. No high-severity vulnerabilities were found; residual risks are mainly design and operational trade-offs managed by the project team. Overall risk is rated **Low**.

2. Scope

- Source file: `eua.sol`
- Dependencies: Uniswap V2 router/pair interfaces, OpenZeppelin ERC20/SafeMath
- Proxy or external governance contracts were not part of this review

Findings are based on static code analysis and do not cover deployment configuration or runtime parameters.

3. Methodology

1. **Code Walkthrough** – Inspect storage, roles, events, and control flow.
2. **Semantic Analysis** – Examine fee formulas, cooldown enforcement, cost-basis handling, and the `recycle` function.
3. **Security Evaluation** – Look for reentrancy, overflows, privilege abuses, and logic flaws.
4. **Design Review** – Assess economic assumptions and centralisation dependencies.

4. Findings Overview

ID	Title	Severity	Status
L-01	Centralised control over fee routing	**Low**	**Known risk**
L-02	Profit tax depends on approximate cost basis	**Low**	**Known risk**
L-03	Cooldown only restricts sells	**Low**	**For awareness**
L-04	`recycle` can disturb liquidity reserves	**Low**	**For awareness**

> The contract relies on centralised operators for fee allocation, tax logic, and redemption interactions. The assessment assumes the team follows its stated operational commitments.

5. Detailed Findings

L-01 Centralised control over fee routing

Severity: Low

Description Buy and sell operations route fees to various addresses (`specialFeeWallet`, `operationsWallet`, `withdrawTokenContract`), all configurable by `onlyOwner`. The initial owner therefore retains full control over fee percentages and recipients. If the controlling keys are compromised or policies change unexpectedly, funds can be diverted.

```
uint256 public buySpecialFee = 370000; // 37%
uint256 public sellSpecialFee = 370000; // 37%
...
function setFeeStructure(...) external onlyOwner { ... }
function setSpecialFeeWallet(address _specialFeeWallet) external onlyOwner { ...
    }
function setWithdrawTokenContract(address _withdrawTokenContract) external
    onlyOwner { ... }
```

Impact The fee system depends on the project operator's integrity. While this does not create a direct contract exploit, users must trust operational controls.

Recommendation

- Move key fee parameters to a multisig or timelock for better accountability.
- Publicly document fee schedules and recipient addresses to increase transparency.

L-02 Profit tax depends on approximate cost basis

Severity: Low

Description `_handleSellTransactionWithTax` calculates whether a sale is profitable by comparing the on-chain `userCostBasis` with the simulated USDT proceeds. Cost basis is updated only during buys via `_calculateUSDTCost`, which approximates the spend using current reserves. Discrepancies can arise when:

- Tokens are received via transfers or airdrops (no cost basis update).
- Multiple buys occur within the cooldown window, depending on fluctuating reserves.
- Reserves are thin, leading to unstable calculations.

```
userCostBasis[buyer] = userCostBasis[buyer].add(usdtCost);
...
if (currentCostBasis >= usdtReceived) {
    userCostBasis[seller] = currentCostBasis.sub(usdtReceived);
} else if (currentCostBasis > 0 && currentCostBasis < usdtReceived) {
    uint256 profitUSDT = usdtReceived.sub(currentCostBasis);
    uint256 profitTokenAmount = _calculateTokensFromUSDTProfit(profitUSDT);
    taxAmount = profitTokenAmount.mul(profitTaxRate).div(100000);
} else {
    taxAmount = tokenAmount.mul(profitTaxRate).div(100000);
}
```

Impact Tax calculations can deviate from actual gains or losses. This primarily affects user experience rather than contract safety, but accurate off-chain monitoring is required.

Recommendation

- Document that profit tracking is approximate and depends on liquidity conditions.
- Enhance off-chain monitoring and be prepared to adjust or reset cost basis entries when anomalies arise.

L-03 Cooldown only restricts sells

Severity: Low

Description The contract records `lastBuyTime` during buys and enforces `block.timestamp >= lastBuyTime + cooldownTime` (default 1 minute) before a user can sell. It does not prevent additional buys during the cooldown, nor does it restrict existing holders from selling immediately if `lastBuyTime` is unset.

```
lastBuyTime[buyer] = uint40(block.timestamp);
...
require(block.timestamp >= lastBuyTime[seller] + cooldownTime, "Cooldown period not met");
```

Impact The cooldown mitigates basic sandwich attacks but does not block pre-existing holders or coordinated strategies. It is a partial defence that should be communicated to users.

Recommendation

- Keep the cooldown but clarify its limitations.
- Consider combining it with slippage limits or per-transaction caps for additional protection.

L-04 `recycle` can disturb liquidity reserves

Severity: Low

Description recycle lets authorised redemption contracts withdraw up to one-third of the pair reserves and transfer them back to the caller, then sync the pair. This reduces pool liquidity and can temporarily increase the token price.

```
function recycle(uint256 amount) external {
    require(isRedemptionContract[msg.sender], "cycle");
    uint256 maxBurn = balanceOf(uniswapV2Pair) / 3;
    uint256 burnAmount = amount >= maxBurn ? maxBurn : amount;
    super._transfer(uniswapV2Pair, msg.sender, burnAmount);
    IUniswapV2Pair(uniswapV2Pair).sync();
}
```

Impact Redemption contracts hold strong influence over market pricing and must operate responsibly.

Recommendation

- Rate-limit or monitor recycle usage on-chain.
- Publish operational policies so users understand how this mechanism affects market dynamics.

6. Architecture and Design Review

- The contract cleanly extends OpenZeppelin ERC20, with custom logic concentrated in _transfer and auxiliary functions.
- Fee routing, cooldowns, and cost basis tracking are intertwined; comprehensive testing is advised.
- Administrative actions rely on onlyOwner; migrating to multisig/timelock arrangements is recommended once deployed.
- swapAndLiquify uses amountOutMin = 0, leaving LP operations exposed to slippage; operational monitoring is needed.

7. Systemic Risk Assessment

- **Governance:** The owner can reconfigure key wallets and fees; long-term operation should move to shared control.
- **Market Dynamics:** High fee percentages and tax mechanics require clear communication to users.

- **Cooldown Effectiveness:** The sell cooldown helps but does not eliminate all sandwich or manipulation vectors.
- **Accounting Accuracy:** Profit tax depends on off-chain assumptions and liquidity conditions.

Overall, risks stem from operational practices and centralised components rather than exploitable code. With proper governance and monitoring in place, the contract is considered **Low** risk.

8. Recommendations and Next Steps

1. Publish multisig or timelock arrangements for fee and wallet management.
2. Monitor cost basis movements and `recycle` activity, triggering alerts for anomalies.
3. Evaluate adding slippage constraints or automated alerts to liquidity operations.
4. Maintain comprehensive test coverage for fee paths, cooldown logic, cost-basis edge cases, and `recycle` limits.

Disclaimer

This report is for informational purposes and does not constitute investment advice. The assessment is based on the provided source code at a specific point in time and is not an endorsement of the project or its team. Smart contracts carry inherent risks; users should perform independent due diligence before interacting with any protocol. Residual risks may remain, and no guarantees are given against undiscovered vulnerabilities.