QuillAudits

Audit Report August, 2022

For

Table of Content

Executive Summary

Project Name	EURB
Overview	The token is intended to be a stable coin that is highly convenient. Following the ERC-20 and BEP-20 protocols and built on both the Ethereum blockchain and Binance Smart Chain, EURB can be sent to or received by anyone with an Ethereum and/or BSC wallet, with no human error based on the smart contract, and participates in the larger global token community.
Timeline	19 August,2022 to 23 August,2022
Method	Manual Review, Functional Testing, Automated Testing, etc.
Scope of Audit	The scope of this audit was to analyse- 1)excludeFromFee 2)IncludeInFee 3) excludeReceiverFromFee 4) IncludeReceiverInFee 5) isTransactionExcludedFromFetfrom 6) getTransactionFee 7)transferFee In EURB codebase for quality, security, and correctness. *https://github.com/RevenyouIO/eurb/tree/master* **Commit hash**: 2ee41f81afce977c77f396f7a5a234a5f795e0c3
Fixed In	*https://github.com/RevenyouIO/eurb/blob/13413b01b364c9dd81c8d4177475ad9542ebe204/contracts/EURB.sol* **Commit hash:** 1d097d47c4b35a9aea742fb778099fdbb7591f63

Executive Summary

3
Issues Found

🟥 High 🟨 Medium

🟩 Low 🟪 Informational

	High	Medium	Low	Informational
Open Issues	0	0	0	0
Acknowledged Issues	0	0	0	3
Partially Resolved Issues	0	0	0	0
Resolved Issues	0	0	0	0

Types of Severities

High

A high severity issue or vulnerability means that your smart contract can be exploited. Issues on this level are critical to the smart contract's performance or functionality, and we recommend these issues be fixed before moving to a live environment.

Medium

The issues marked as medium severity usually arise because of errors and deficiencies in the smart contract code. Issues on this level could potentially bring problems, and they should still be fixed.

Low

Low-level severity issues can cause minor impact and or are just warnings that can remain unfixed for now. It would be better to fix these issues at some point in the future.

Informational

These are severity issues that indicate an improvement request, a general question, a cosmetic or documentation error, or a request for information. There is low-to-no impact.

Types of Issues

Open

Security vulnerabilities identified that must be resolved and are currently unresolved.

Resolved

These are the issues identified in the initial audit and have been successfully fixed.

Acknowledged

Vulnerabilities which have been acknowledged but are yet to be resolved.

Partially Resolved

Considerable efforts have been invested to reduce the risk/impact of the security issue, but are not completely resolved.

Checked Vulnerabilities

✓ Re-entrancy	✓ Tautology or contradiction
✓ Timestamp Dependence	✓ Return values of low-level calls
✓ Gas Limit and Loops	✓ Missing Zero Address Validation
✓ Exception Disorder	✓ Private modifier
✓ Gasless Send	✓ Revert/require functions
✓ Use of tx.origin	✓ Using block.timestamp
✓ Compiler version not fixed	✓ Multiple Sends
✓ Address hardcoded	✓ Using SHA3
✓ Divide before multiply	✓ Using suicide
✓ Integer overflow/underflow	✓ Using throw
✓ Dangerous strict equalities	✓ Using inline assembly

Techniques and Methods

Throughout the audit of smart contract, care was taken to ensure:

- The overall quality of code.
- Use of best practices.
- Code documentation and comments match logic and expected behaviour.
- Token distribution and calculations are as per the intended behaviour mentioned in the whitepaper.
- Implementation of ERC-20 token standards.
- Efficient use of gas.
- Code is safe from re-entrancy and other vulnerabilities.

The following techniques, methods and tools were used to review all the smart contracts.

Structural Analysis

In this step, we have analysed the design patterns and structure of smart contracts. A thorough check was done to ensure the smart contract is structured in a way that will not result in future problems.

Static Analysis

Static analysis of smart contracts was done to identify contract vulnerabilities. In this step, a series of automated tools are used to test the security of smart contracts.

Code Review / Manual Analysis

Manual analysis or review of code was done to identify new vulnerabilities or verify the vulnerabilities found during the static analysis. Contracts were completely manually analysed, their logic was checked and compared with the one described in the whitepaper. Besides, the results of the automated analysis were manually verified.

Gas Consumption

In this step, we have checked the behaviour of smart contracts in production. Checks were done to know how much gas gets consumed and the possibilities of optimization of code to reduce gas consumption.

Tools and Platforms used for Audit

Remix IDE, Truffle, Truffle Team, Solhint, Mythril, Slither, Solidity statistic analysis.

Manual Testing

A. Contract - EURB

High Severity Issues

No issues found

Medium Severity Issues

No issues found

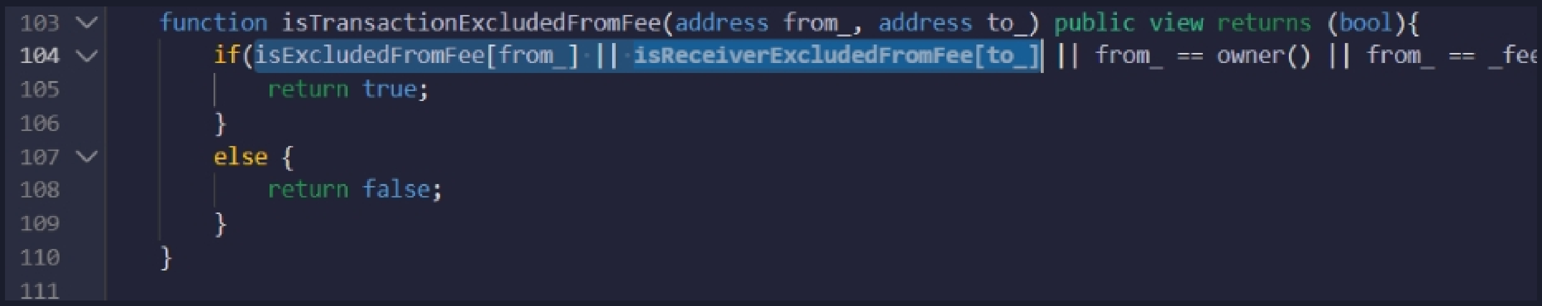
Low Severity Issues

No issues found

Informational Issues

1. One mapping can be used to store excluded accounts

Description
The transaction is excluded or not is getting checked in isTransactionExcludedFromFee() with the if condition. While checking it checks that if addresses are set to true in isExcludedFromFee and isReceiverExcludedFromFee mappings. Here, only one mapping can be used instead of two to store excluded transactions which can in turn reduce the requirement of adding additional functions to exclude/incude addresses from other mapping.



```
    function isTransactionExcludedFromFee(address from_, address to_) public view returns (bool){
        if(isExcludedFromFee[from_] || isReceiverExcludedFromFee[to_] || from_ == owner() || from_ == _fee
            return true;
        }
        else {
            return false;
        }
    }
```

Remediation
Consider reviewing logic and storing excluded addresses in one mapping.

Status
Acknowledged

2. Some functions lack return values

Description
Some functions e.g excludeReceiverFromFee(), includeReceiverInFee(), excludeSenderFromFee, includeSenderInFee) are not returning bool values as mention in the screenshot shared with our team internally.

Remediation
Review the code and add return bool values if required.

Status

Acknowledged

3. Centralization issue

Description
A malicious owner can set 100% fee on transfer amount using setFeePercentage() because the require statement checks for “feePercentage_ <= 100000” that means feePercentage_ can be set 100%. Which can deduct the whole transfer amount as a fee.

Remediation
Consider checking the entered feePercentage_ would be less than 100% . Which will prevent from setting a 100% fee.

Status

Acknowledged

Functional Tests

- ✓ Should be able to grant Minter, Burner and Asset Protection Roles to accounts.
- ✓ Should be able to Mint and Burn tokens (from the owner's account as well as from any other account)
- ✓ Should be able to transfer tokens
- ✓ Should be able to transfer ownership and revert for a zero address.
- ✓ Should revert if transfer amount exceeds balance
- ✓ Should revert if Minter and Burners don't have desired roles.
- ✓ Should be able to set a fee receiver account
- ✓ Should be able to deduct the fee from the transfer amount and transfer it to the fee recipient.
- ✓ Should not deduct fee if sender or receiver is owner,fee recipient, excluded from fee, or the fee percentage amount is not greater than zero.
- ✓ Should be able to freeze and unfreeze accounts and revert if the caller does not have the asset protection role
- ✓ Owner should be able to stop minting, burning and pause/unpause the contract.
- ✓ Owner should be able to revoke the role
- ✓ Should revert if asset protection role tries to freeze owner account

Automated Tests

No major issues were found. Some false positive errors were reported by the tools. All the other issues have been categorized above according to their level of severity.

Closing Summary

In this report, we have considered the security of the EURB Contract. We performed our audit according to the procedure described above.

Some issues of informational severity were found. Some suggestions and best practices are also provided in order to improve the code quality and security posture.

Disclaimer

QuillAudits smart contract audit is not a security warranty, investment advice, or an endorsement of the EURB Platform. This audit does not provide a security or correctness guarantee of the audited smart contracts.

The statements made in this document should not be interpreted as investment or legal advice, nor should its authors be held accountable for decisions made based on them. Securing smart contracts is a multistep process. One audit cannot be considered enough. We recommend that the EURB Team put in place a bug bounty program to encourage further analysis of the smart contract by other third parties.

About QuillAudits

QuillAudits is a secure smart contracts audit platform designed by QuillHash Technologies. We are a team of dedicated blockchain security experts and smart contract auditors determined to ensure that Smart Contract-based Web3 projects can avail the latest and best security solutions to operate in a trustworthy and risk-free ecosystem.

500+	**$15B**	**500K**
Audits Completed	Secured	Lines of Code Audited

Follow Our Journey

Audit Report August, 2022

For

QuillAudits

Canada, India, Singapore, United Kingdom

audits.quillhash.com

audits@quillhash.com