QuillAudits

Audit Report
April, 2022

For

Table of Content

Executive Summary

Project Name	EURB Smart Contracts By RevenYou
Overview	The EURB 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	April 1, 2022 - April 8, 2022
Method	Manual Review, Functional Testing, Automated Testing etc.
Audit Scope	The scope of this audit was to analyse EURB smart contract's codebase for quality, security, and correctness. Master
Source Code	https://github.com/RevenyouIO/eurb
Commit Hash	1190d65c5bab3dc45c479e7f9454669438408582
Fixed In	32798ba60080256d61d777d74fd7ae8b89930925

High Medium

Low Informational

	High	Medium	Low	Informational
Open Issues	0	0	0	0
Acknowledged Issues	0	0	0	0
Partially Resolved Issues	0	0	0	0
Resolved Issues	0	1	1	3

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
- Timestamp Dependence
- Gas Limit and Loops
- DoS with Block Gas Limit
- Transaction-Ordering Dependence
- Use of tx.origin
- Exception disorder
- Gasless send
- Balance equality
- Byte array
- Transfer forwards all gas
- ERC20 API violation
- Malicious libraries
- Compiler version not fixed
- Redundant fallback function
- Send instead of transfer
- Style guide violation
- Unchecked external call
- Unchecked math
- Unsafe type inference
- Implicit visibility leve

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 were found

Medium Severity Issues

A.1 Broken Access Control

Line	Function - freezeAccount

268	`function freezeAccount(address _account) public {`
	`  require(`
	`  hasRole(ASSET_PROTECTION_ROLE, _msgSender()),`
	`  "EURB: Caller does not have asset protection role"`
	`  );`
	`  require(isFrozen[_account] != true, "EURB: Already froze account");`
	`  isFrozen[_account] = true;`
	`}`

Description

The accounts having the asset protection role have permissions to even freeze the owner’s account that could lead to lack of transfer of tokens from the owner's account and put the hierarchy of authority in question. Moreover, the accounts with the asset protection role can stop the owner from transferring tokens any time they please.

Remediation

The accounts must be verified for authority before granting the Asset protection role and they should not have the authority to freeze the owner’s account.

Status

Fixed

Low Severity Issues

A.2 Missing check for feePercentage

Line	Function - transferFee

335	`function transferFee(address sender, address recipient, uint256 amount) internal {`
	`  uint256 txFee = 0;`
	`  if(sender != owner() && sender != _feeReceiver && !isExcludedFromFee[sender]  && _feePercentage > 0) {`
	`  txFee = amount * _feePercentage / 100000;`
	`  _transfer(sender, _feeReceiver, txFee);`
	`  }`
	`  _transfer(sender, recipient, amount - txFee);`
	`}`

Description

While calculating the transaction fee, there is no check for the _feePercentage that could be easily set to 100 and that would result in transferring the whole transfer amount to the fee Receiver's account and the receiver will get zero tokens

Remediation

To solve the issue, a check should be placed in the function that makes sures that the feePercentage is always less than 1,00,000.

Status

Fixed

Informational Issues

A.3 Misleading error message

Line	Function - burn

173	`function burn(uint256 amount) public override canBurn whenNotPaused {`
	`  require(`
	`  hasRole(BURNER_ROLE, _msgSender()),`
	`  "EURB: Caller is not a burner nor has asset protection role"`
	`  );`
	`  _burn(owner(), amount);`
	`  emit Burn(owner(), amount);`
	`}`

Description

In the burn function the check is to see whether the caller has a burner role or not but the error message says “Caller is not a burner nor has asset protection role” as there is no check for the “Asset Protection” role, it may mislead the user and could be interpreted that an account should have both the roles (burner and asset protection) in order to burn tokens.

Remediation

There could be two fixes depending on the business logic, if it’s intended that in order to burn tokens, an account should have both the roles then there should be a check for the “Asset Protection” role and if not then the error message should be changed.

Status

Fixed

A.4 Unlocked pragma (pragma solidity ^0.8.0)

Description

Contracts should be deployed with the same compiler version and flags that they have been tested with thoroughly. Locking the pragma helps to ensure that contracts do not accidentally get deployed using, for example, an outdated compiler version that might introduce bugs that affect the contract system negatively.

Remediation

Here all the in-scope contracts have an unlocked pragma, it is recommended to lock the same. Moreover, we strongly suggest not to use experimental Solidity features (e.g., pragma experimental ABIEncoderV2) or third-party unaudited libraries. If necessary, refactor the current code base to only use stable features.

Status

Fixed

A.5 General Recommendation

In the case of stable coins, the hierarchy of authority plays a crucial role and our team recommends that the roles with privileges must be given to the accounts with proven authority and functions like “finishMinting” and “finishBurning” must be used cautiously because once these functions are called on the mainnet then could be no minting or burning. Hence, it will be an irreversible change

Status

Fixed

B. Contract - Proxy

High Severity Issues

No issues were found

Medium Severity Issues

No issues were found

Low Severity Issues

No issues were found

Informational Issues

No issues were found

Functional Testing

Some of the tests performed are mentioned below

- ✓ 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 is owner, receiver is 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.

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 Smart Contract by Revenyou. We performed our audit according to the procedure described above.

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

At the End, Revenyou Team Fixed all Issues.

Disclaimer

QuillAudits smart contract audit is not a security warranty, investment advice, or an endorsement of the Revenyou 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 Revenyou 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+
Audits Completed

$15B
Secured

500K
Lines of Code Audited

Follow Our Journey

Audit Report
April, 2022

For

QuillAudits

Canada, India, Singapore, United Kingdom

audits.quillhash.com

audits@quillhash.com