PROOF

BEEP SMART CONTRACT SECURITY AUDIT REPORT

Disclaimer

This is a limited report of findings based on an analysis of industry best practices as of the date of this report regarding cybersecurity vulnerabilities and issues in smart contract frameworks and algorithms, the details of which are detailed in this report. stated in the report. To get the full picture of our analysis, it's important to read the full report. Although we have conducted our analysis and have done our best to prepare this report, you should not rely on this report and cannot claim against us based on what it does or does not say or how it was produced. It is important to do your own research before making any decisions. This is explained in more detail in the following disclaimer. Please be sure to read to the end.

Disclaimer:

BY READING THIS REPORT OR ANY PART THEREOF, YOU AGREE TO THE TERMS OF THIS DISCLAIMER. If you do not agree to the terms, then please immediately cease reading this report, and delete and destroy any and all copies of this report downloaded and/or printed by you. This report is provided for information purposes only and on a non-reliance basis, and does not constitute investment advice. No one shall have any right to rely on the report or its contents, and Proof Audit and its affiliates (including holding companies, shareholders, subsidiaries, employees, directors, officers and other representatives) (Proof Audit) owe no duty of care towards you or any other person, nor does Proof Audit make any warranty or representation to any person on the accuracy or completeness of the report. The report is provided "as is", without any conditions, warranties or other terms of any kind except as set out in this disclaimer, and Proof Audit hereby excludes all representations, warranties, conditions and other terms (including, without limitation, the warranties implied by law of satisfactory quality, fitness for purpose and the use of reasonable care and skill) which, but for this clause, might have effect in relation to the report. Except and only to the extent that it is prohibited by law, Proof Audit hereby excludes all liability and responsibility, and neither you nor any other person shall have any claim against Proof Audit, for any amount or type of loss or damage that may result to you or any other person (including without limitation, any direct, indirect, special, punitive, consequential or pure economic loss or damages, or any loss of income, profits, goodwill, data, contracts, use of money, or business interruption, and whether in delict, tort (including without limitation negligence), contract, breach of statutory duty, misrepresentation (whether innocent or negligent) or otherwise under any claim of any nature whatsoever in any jurisdiction) in any way arising from or connected with this report and the use, inability to use or the results of use of this report, and any reliance on this report.

Security analytics are based solely on smart contracts. Application or process security not checked. Product code not reviewed.

PROOF

Table of Contents

Executive Summary

Objectives

Proof Audit carried out an audit of BEEP, specifically their ERC20 token. The project is based on the Ethereum Network. We reviewed documentation which helped with understanding the functions of their code. Our findings in the audit ranged from minor to critical.

Project Info

Audited project

BEEP

Deployer Address

0xdd1c8C22593ec7dd8AaCd96F914550e0D3D568FB

Contract Address

0x0283d310d682284EbC24DB33A41Bb5a01BDD140B

Blockchain

Ethereum

Project website:

https://www.beepeth.com/

Methodology

During the audit process, we inspected the repository thoroughly, using a line-by-line code read through to review vulnerabilities, quality of the code and adherence to best practices and specifications. We used Computer-Aided Verification to support the audit process.

Our auditing process is as follows:

1. Code Review:
   A review of the scope, specifications and documentation provided to ensure an in depth understanding of the purpose and functionality of the relevant smart contracts.
2. Automated Analysis:
   A series of reviews carried out with the use of automated tools. These reviews serve as a basis for further manual analysis and provide relevant visualizations of the code.
3. Testing & Manual Review of Code:
   Test coverage analysis and a line-by-line read through of the project code in order to identify vulnerabilities, errors and weaknesses in code quality.
4. Specification Comparison:
   A review of the code against the specifications provided to ensure that the code operates as is intended.
5. Best Practices Review:
   A review of the smart contracts to identify potential improvements in effectiveness, efficiency, and maintainability, with a focus on adherence to industry best practices.

Scope

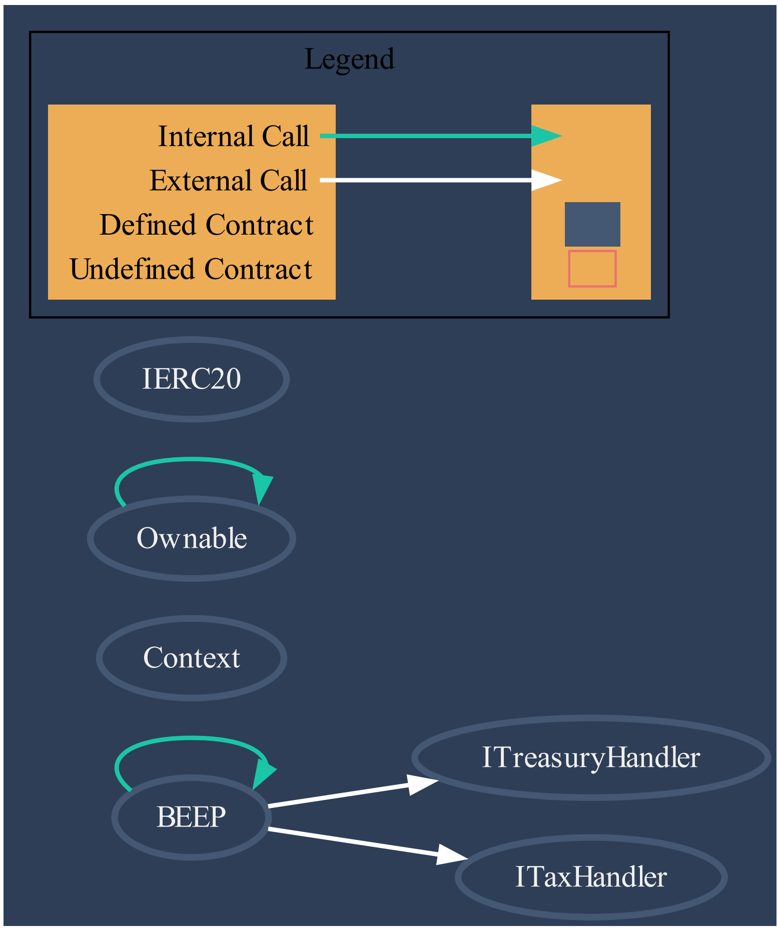
The contracts audited are from
https://etherscan.io/token/0x0283d310d682284ebc24db33a41bb5a01bdd140b?a=0xaa1656b7d4629476fa4cf76ccfbc01a4653bac71#code

The audited contracts are:

BEEP.sol

The scope of the audit is limited to these files. No other files in this repository were audited. Its dependencies are assumed to work according to their documentation. Also, no tests were reviewed for this audit.

BEEP.sol Interaction Graph



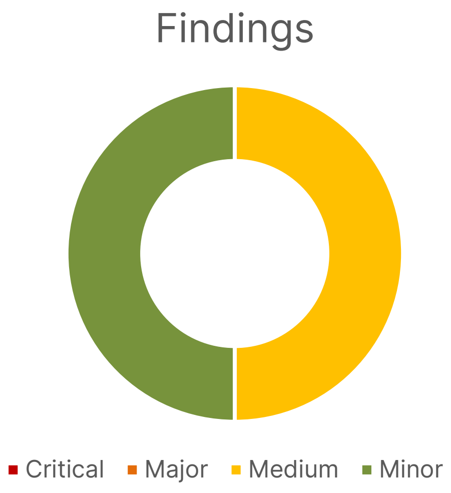
Analyses

Without being limited to them, the audit process included the following analyses:

- Arithmetic errors
- Outdated version of Solidity compiler
- Race conditions
- Reentrancy attacks
- Misuse of block timestamps
- Denial of service attacks
- Excessive gas usage
- Missing or misused function qualifiers
- Needlessly complex code and contract interactions
- Poor or nonexistent error handling
- Insufficient validation of the input parameters
- Incorrect handling of cryptographic signatures
- Centralization and upgradeability

Summary of Findings

We found **0** critical issue, **0** Major issues, **1** medium issues, and **1** minor issues.



Security Issues

ID	Title	Severity	Status
01	Reentrancy in BEEP._transfer	Medium	Acknowledged
02	A floating Pragma is set.	Minor	Acknowledged

Findings

Severity Classification

Security risks are classified as follows:

- **Critical:** These are issues that we manage to exploit. They compromise the system seriously. They must be fixed **immediately**.
- **Medium:** These are potentially exploitable issues. Even though we did not manage to exploit them, or their impact is not clear, they might represent a security risk in the near future. We suggest fixing them **as soon as possible**.
- **Minor:** These issues represent problems that are relatively small or difficult to take advantage of but can be exploited in combination with other issues. These kinds of issues do not block deployments in production environments. They should be taken into account and be fixed **when possible**.

Issues Status

An issue detected by this audit can have four distinct statuses:

- **Unresolved**: The issue has not been resolved.
- **Acknowledged**: The issue remains in the code but is a result of an intentional decision.
- **Resolved**: Adjusted program implementation to eliminate the risk.
- **Partially resolved**: Adjusted program implementation to eliminate part of the risk. The other part remains in the code but is a result of an intentional decision.
- **Mitigated**: Implemented actions to minimize the impact or likelihood of the risk.

Critical Severity Issues

N/A

Major Severity Issues

N/A

Medium Severity Issues

Reentrancy in BEEP._transfer

Description: There is a risk of a reentrancy vulnerability in the _transfer function.

External calls:

- treasuryHandler.beforeTransferHandler(from,to,amount)

State variables written after the call(s):

- _balances[from] -= amount
- _balances[to] += taxedAmount
- _balances[address(treasuryHandler)] += tax

Recommendation: We recommend the client the “checks, effects, interactions” pattern to reduce the attack surface for malicious contracts trying to hijack control flow after an external call.

Status: **Acknowledged**

Minor Severity Issues

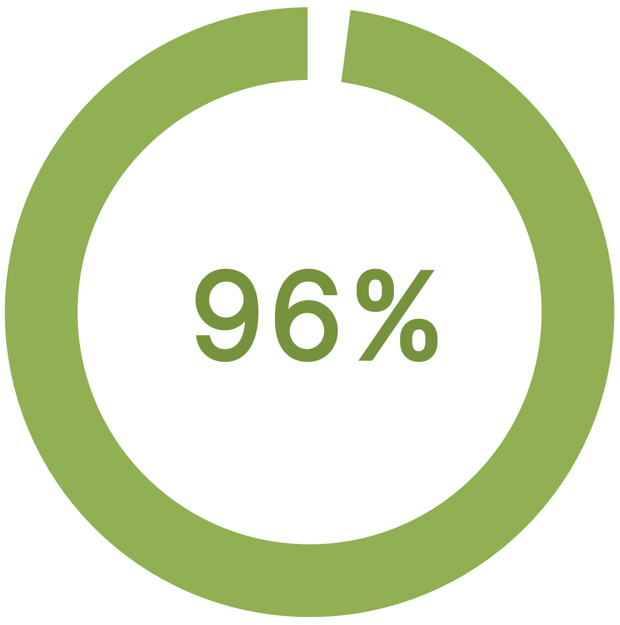
A floating Pragma is set

Description: The current pragma Solidity directive is ""^0.8.0"".

Recommendation: It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

Status: **Acknowledged**

Security Rating



Based on Vulnerabilities Found