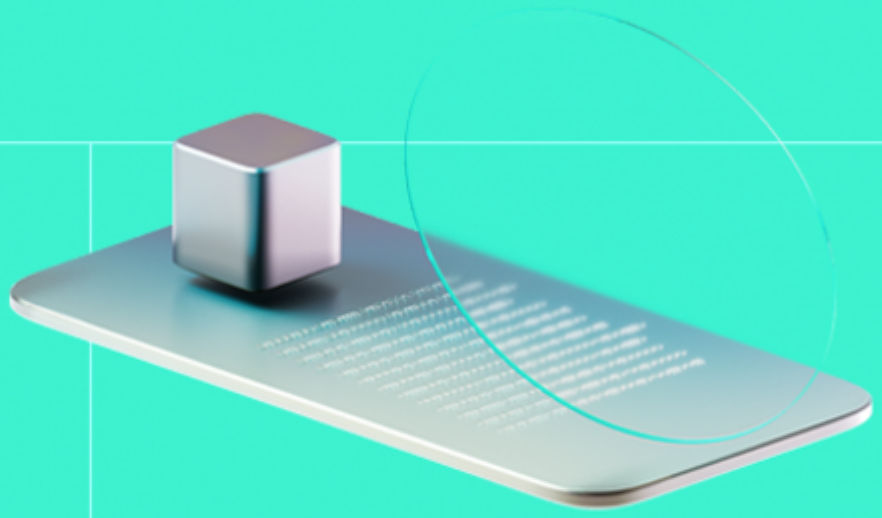
HACKEN

Smart Contract Code Review And Security Analysis Report



Customer: Credbull

Date: 16/10/2024

We express our gratitude to the Credbull team for the collaborative engagement that enabled the execution of this Smart Contract Security Assessment.

Credbull is an integrated Decentralized Finance (DeFi) platform giving investors access to private credit as an asset class. The audit particularly focuses on Credbull CBL token.

Document

Name	Smart Contract Code Review and Security Analysis Report for Credbull
Audited By	Stepan Chekhovskoi
Approved By	Ataberk Yavuzer
Website	https://credbull.io
Changelog	16/10/2024 - Preliminary Report
Platform	Arbitrum
Language	Solidity
Tags	ERC-20
Methodology	https://hackenio.cc/sc_methodology

Review Scope

Repository	https://github.com/credbull/credbull-defi
Commit	0792b613b5253440ccd181f5614403ec6382da81

Audit Summary

The system users should acknowledge all the risks summed up in the risks section of the report

0	0	0	0
Total Findings	Resolved	Accepted	Mitigated

Findings by Severity

Severity	Count
Critical	0
High	0
Medium	0
Low	0

Documentation quality

- The contract code is covered with comprehensive documentation and NatSpec comments.
- The repository provides instructions for development environment setup.

Code quality

- The code is clean and effectively utilizes standard ERC-20 token implementation with some extensions.

Test coverage

Code coverage of the project is **100%** (branch coverage).

- The code is covered with tests.

Table of Contents

System Overview

The CBL contract is ERC20-compliant token with additional features listed below.

- **Permit:** Allows approvals to be made via signatures, following the EIP-712 standard.
- **Mintable**: Authorized users are allowed to mint additional tokens.
- **Burnable:** Tokens can be burned, reducing the total supply.
- **Capped Supply:** The total supply of tokens is capped.

The token properties are set as follows.

- **Name**: Credbull.
- **Symbol**: CBL.
- **Max Supply**: Set at the moment of deployment.

Privileged roles

- The specified minters are allowed to mint tokens until the supply cap is not exceeded.

Potential Risks

- **Centralized Minting to a Single Address**: The project concentrates minting tokens in a single address, raising the risk of fund mismanagement or theft, especially if key storage security is compromised.
- **Single Points of Failure and Control**: The project is fully or partially centralized, introducing single points of failure and control. This centralization can lead to vulnerabilities in decision-making and operational processes, making the system more susceptible to targeted attacks or manipulation.
- **Burned Tokens Remint Possibility**: The token implements Burnable and Capped Supply extensions, allowing authorized minters to remint tokens that were burned. Such behavior might be unclear to users, as burning usually means a decrease in circulating supply. In this case, it may not function this way, especially if minters are not centralized (for example, mint ability delegated to another smart contract).
- **ERC-20 Permit Functionality**: The permit functionality poses a risk of race conditions between authorizations with the same nonce. The permit implementation requires all authorizations to be applied sequentially, with each nonce increased by one.

Findings

Vulnerability Details

No security vulnerabilities were detected in the contracts during the audit.

Observation Details

F-2024-6659 - Floating Pragma - Info

Description: The contracts use floating pragma `^0.8.20`.

This may result in the contracts being deployed using the wrong pragma version, which is different from the one they were tested with. For example, the contracts might be deployed using an outdated pragma version which may include bugs that affect the system negatively.

Assets:

- File: token/CBL.sol [https://github.com/credbull/credbull-defi/]

Status: Pending Fix

Recommendations

Remediation: Consider locking the pragma version whenever possible and avoid using a floating pragma in the final deployment. Consider known bugs for the compiler version that is chosen.

Disclaimers

Hacken Disclaimer

The smart contracts given for audit have been analyzed based on best industry practices at the time of the writing of this report, with cybersecurity vulnerabilities and issues in smart contract source code, the details of which are disclosed in this report (Source Code); the Source Code compilation, deployment, and functionality (performing the intended functions).

The report contains no statements or warranties on the identification of all vulnerabilities and security of the code. The report covers the code submitted and reviewed, so it may not be relevant after any modifications. Do not consider this report as a final and sufficient assessment regarding the utility and safety of the code, bug-free status, or any other contract statements.

While we have done our best in conducting the analysis and producing this report, it is important to note that you should not rely on this report only — we recommend proceeding with several independent audits and a public bug bounty program to ensure the security of smart contracts.

English is the original language of the report. The Consultant is not responsible for the correctness of the translated versions.

Technical Disclaimer

Smart contracts are deployed and executed on a blockchain platform. The platform, its programming language, and other software related to the smart contract can have vulnerabilities that can lead to hacks. Thus, the Consultant cannot guarantee the explicit security of the audited smart contracts.

Appendix 1. Definitions

Severities

When auditing smart contracts, Hacken is using a risk-based approach that considers **Likelihood**, **Impact**, **Exploitability** and **Complexity** metrics to evaluate findings and score severities.

Reference on how risk scoring is done is available through the repository in our Github organization:

hknio/severity-formula

Severity	Description
Critical	Critical vulnerabilities are usually straightforward to exploit and can lead to the loss of user funds or contract state manipulation.
High	High vulnerabilities are usually harder to exploit, requiring specific conditions, or have a more limited scope, but can still lead to the loss of user funds or contract state manipulation.
Medium	Medium vulnerabilities are usually limited to state manipulations and, in most cases, cannot lead to asset loss. Contradictions and requirements violations. Major deviations from best practices are also in this category.
Low	Major deviations from best practices or major Gas inefficiency. These issues will not have a significant impact on code execution.

Potential Risks

The "Potential Risks" section identifies issues that are not direct security vulnerabilities but could still affect the project’s performance, reliability, or user trust. These risks arise from design choices, architectural decisions, or operational practices that, while not immediately exploitable, may lead to problems under certain conditions. Additionally, potential risks can impact the quality of the audit itself, as they may involve external factors or components beyond the scope of the audit, leading to incomplete assessments or oversight of key areas. This section aims to provide a broader perspective on factors that could affect the project's long-term security, functionality, and the comprehensiveness of the audit findings.

Appendix 2. Scope

The scope of the project includes the following smart contracts from the provided repository:

Scope Details	
Repository	https://github.com/credbull/credbull-defi
Commit	0792b613b5253440ccd181f5614403ec6382da81
Whitepaper	https://docs.credbull.io/docs/litepaper
Requirements	packages/contracts/test/src/token/CBL_README.md
Technical Requirements	README.md, packages/contracts/README.md

Asset	Type
File: token/CBL.sol [https://github.com/credbull/credbull-defi/]	Smart Contract

Appendix 3. Additional Valuables

Additional Recommendations

The smart contracts in the scope of this audit could benefit from the introduction of automatic emergency actions for critical activities, such as unauthorized operations like ownership changes or proxy upgrades, as well as unexpected fund manipulations, including large withdrawals or minting events. Adding such mechanisms would enable the protocol to react automatically to unusual activity, ensuring that the contract remains secure and functions as intended.

To improve functionality, these emergency actions could be designed to trigger under specific conditions, such as:

- Detecting changes to ownership or critical permissions.
- Monitoring large or unexpected transactions and minting events.
- Pausing operations when irregularities are identified.

These enhancements would provide an added layer of security, making the contract more robust and better equipped to handle unexpected situations while maintaining smooth operations.