CPRO Token Audit Report

Document version: **1.0.1**

Auditors:
Srdjan Delić
Tatjana Radovanović

Audit date:
27.10.2025

Contents:

Overview

Repository / commit hash / tag: _____
Files / folders included in the audit: CPROToken.sol
Language / compiler version: Solidity ^0.8.28
Testing networks: Remix, Sepolia, ChatGPT
Audit start date: 2025-10-27
Audit end date: 2025-10-27

About CPRO

CPRO (CryptoProccessor) Token is an ERC-20 standard token developed by the CPRO Team for the Ethereum chain.

Disclaimer

The audit team will implement extensive and exhaustive internal tooling and methodologies into providing a complete token audit process. The audit team **cannot** guarantee that all potential vulnerabilities or risks will be identified and as such holds **no responsibility** if issues are later identified in any further step of the development or deployment process for the token smart contract(s) being audited.

The audit is focused on Solidity contracts security and quality of code with the consideration for the implementation and adherence of the best, industry-wide practices implemented in the said smart contracts.

In case of future smart contact revisions or edits, all previous audits will be deemed as unverified/deprecated and **invalidated**. For even minor edits to the audited contracts, the same audit methodologies should be applied again in order to achieve the same level of confidence and security.

Risk Classification

Severity chart:

- **Critical** - High impact and high likelihood of happening. Requires immediate attention and fixing
- **High** - Medium impact and high likelihood of happening. Requires attention and fixing
- **Medium** - Medium impacts and medium likelihood of happening. No urgent attention required, but recommended addressing before next deployment/testing cycle

- **Low** - Low impact and low likelihood of happening. Can be postponed to the next development cycle
- **Informative** - Audit team's own suggestions for code,structure and security improvements and potential vulnerabilities that could manifest in the near future, but waren't part of the required token specifications

Tools

Slither - Hardhat/Foundry projects
Aderyn - Hardhat/Foundry projects
AI Tools - ChatGPT, Claude, RemixAI Assistant

Protocol Summary

Description

Documentation

The CPRO project documentation is under an NDA and is not available for public view.

Actors and Roles

Actors

- Token_Team: Token smart contract developers and core team members.
- Users: Receivers of the CPRO token.

Key Components

CPROToken.sol is the minting contract developed by the CPRO Team. It combines standard token functionalities with advanced features like voting, gasless transactions with administrative controls.

Basic features:

- Transfer, approve, allowance checking
- Standard ERC20 balance checking
- Standard ERC20 events (Transfer, Approve)

Ownership controls:

- Only the contract owner can perform administrative functions
- Ownership can be transferable to another address
- Critical functions are protected by the **ownerOnly** modifier

Token supply management:
- Owner can create new tokens
- Owner can destroy tokens from their own balance
- Limited to **1 billion** tokens. No more can ever be minted/created
- Current supply decreases when tokens are burned

Emergency controls:
- Owner can stop all token transfers on emergency situations
- Owner can restore normal operations

Governance and Voting:
- Users can delegate their voting power to others
- Users can delegate themselves to participate in the voting process
- Users can delegate using signatures to avoid gas fees
- System tracks current and history of voting power
- Snapshots for double voting prevention and manipulation
- Can check voting power at any point in the past

Gasless approvals:
- Users can approve token spending using signatures
- Third parties can submit approval transactions
- Uses cryptographic signatures
- No need for separate transaction approvals

Audit Scope

- CPROToken.sol

Result Summary

No scam instructions found in the provided code snippet, but the **recoverERC20** function is dangerous as-is. Fix it with **SafeERC20**, and the contract will be production-ready. The rest of the findings are optimizations.

Findings

1. **Unchecked external call in recoverERC20()**
   Severity: CRITICAL
   Location: CPROToken.sol, function recoverERC20()
   Description: The function uses transfer() without checking the return value, which can fail silently for some tokens. Recovery operations may fail without notification, potentially causing confusion or loss of funds

   IERC20(tokenAddress).transfer(owner(), tokenAmount);

   Recommendation:

   IERC20(tokenAddress).safeTransfer(owner(), tokenAmount);

   CPRO Team: Acknowledged
   Audit Team: Waiting for fix implementation

2. **Redundant MAX_SUPPLY check in mint**
   Severity: MEDIUM
   Location: CPROToken.sol, function mint
   Description: ERC20Capped already enforces the cap

   Recommendation:

   Remove the require(totalSupply() + amount <= MAX_SUPPLY, ...)

   CPRO Team: Acknowledged
   Audit Team: Waiting for fix implementation

3. **Initial mint in constructor**
   Severity: MEDIUM
   Location: CPROToken.sol, function mint
   Description: Hardcoding 1_000_000 * 10 ** decimals()

   Recommendation:

   consider making the value a parameter for flexibility

   CPRO Team: Acknowledged
   Audit Team: Waiting for fix implementation

4. **Missing whenNotPaused on mint/burnFromOwner**
   Severity: MEDIUM
   Location: CPROToken.sol, function mint/burnFromOwner
   Description: Add the modifier to prevent minting/burning while paused

   Recommendation:

   function mint(address to, uint256 amount) external onlyOwner whenNotPaused { ... }

   CPRO Team: Acknowledged
   Audit Team: Waiting for fix implementation

5. **_update override**
   Severity: MEDIUM
   Location: CPROToken.sol, function _update
   Description: The override is correct but unnecessary

   Recommendation:

   Remove the _update function unless you're adding custom logic.

   CPRO Team: Acknowledged
   Audit Team: Waiting for fix implementation

6. **License**
   Severity: MEDIUM
   Location: CPROToken.sol, header license
   Description: Update license

   UNLICENSED means others can't legally reuse/fork your code

   Recommendation:

   Consider MIT or GPL-3.0 licenses or some other more restrictive like Apache 2.0

   CPRO Team: Acknowledged
   Audit Team: Waiting for fix implementation

Security Controls Review

Control	Status	Notes
Access Control	✅	Owner-only for mint, pause, burn, recover
Pause Mechanism	✅	Properly implemented with modifiers
Supply Cap	✅	Enforced via ERC20Capped
Permit (EIP-2612)	✅	Correct inheritance
Voting (ERC20Votes)	✅	Integrated correctly
Reentrancy	✅	No reentrant external calls
Upgradeable	❌	Non-upgradeable (intended)
Overflow / Underflow	✅	Safe under Solidity ≥0.8
Emergency Recovery	✅	Prevents recovery of self tokens

Gas & Performance Observations

- Minor overhead from multiple inheritance layers.
- `pause()` and `unpause()` are low gas cost and rarely used.
- Mint and burn flows are O(1).
- No unnecessary loops or complex state operations detected.