Coingarage: Smart Contract Audit Review

Prepared for: Coingarage

Date: January 23, 2025

Table of Contents

Disclaimer

This document might include private information regarding the Coingarage (the "Company") intellectual property and IT systems, as well as details on potential weaknesses and ways to exploit them.

After all vulnerabilities have been patched, the Company may decide whether to use the secret information in the report internally or to make it public.

Audit Information

Name	Coingarage
Type of Contracts	Token
Platform	Polygon
Language	Solidity
Testing Methods	Static Analysis using a proprietary rule set and AI Analysis, and Manual Review
Website	https://polygonscan.com/address/0x0b258a4ecc4ac7a15fedb882db5d13f6ef23b02f#code
Date	Jan 23, 2025

Executive Summary

AuditBase has completed an audit for Coingarage, an ERC20 token on Polygon implementing maximum supply, recovery, and burn features. The scope of this audit was limited to the token functionality.

Scope

Deployed Contract	https://polygonscan.com/address/0x0b258a4ecc4ac7a15fedb882db5d13f6ef23b02f#code
Documentation	N/A
Unit Tests	N/A
Contracts	PowerfulERC20, TokenRecover, Roles, ServicePayer

Security Definitions

Risk	Description
Critical	Critical flaws can result in the loss of assets or the alteration of data and are typically simple to exploit.
High	High-level vulnerabilities are challenging to attack, but they can have a big impact on how smart contracts work, like giving the public access to essential features.
Medium	Although medium-level vulnerabilities should be fixed, they cannot result in the loss of assets or the manipulation of data.
Low	Low-level flaws are typically caused by bits of unneeded, old code that don't have a big influence on the execution.

Findings

Centralization risk for trusted owners

Having a single EOA as the only owner of contracts is a large centralization risk and a single point of failure. A single private key may be taken in a hack, or the sole holder of the key may become unable to retrieve the key when necessary. Consider changing to a multi-signature setup, or having a role-based authorization model.

Severity: Medium

```
Java
1512
1521 function renounceOwnership() public virtual onlyOwner {
function transferOwnership(address newOwner) public virtual
onlyOwner {
1539 function recoverERC20(address tokenAddress, uint256
tokenAmount) public virtual onlyOwner {
1595 function _mint(address account, uint256 amount) internal
override(ERC20, ERC20Capped) onlyMinter {
1604 function _finishMinting() internal override onlyOwner {
```

Use `Ownable2Step` rather than `Ownable`

`Ownable2Step` and `Ownable2StepUpgradeable` prevent the contract ownership from mistakenly being transferred to an address that cannot handle it (e.g. due to a typo in the address), by requiring that the recipient of the owner permissions actively accept via a contract call of its own.

Severity: Low

```
Java
File: contract.sol

1533 contract TokenRecover is Ownable {
```

`_setupRole` is Deprecated

`_setupRole` is deprecated in favor of `grantRole` and `renounceRole`.

Severity: Low

```
Java
File: contract.sol

function _setupRole(bytes32 role, address account) internal virtual {
_setupRole(DEFAULT_ADMIN_ROLE, _msgSender());
_setupRole(MINTER_ROLE, _msgSender());
```

Disclaimers

AuditBase Disclaimer

The smart contracts given for audit have been analyzed by the best industry practices at the date 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).

Regarding the code's security, the audit offers no claims or guarantees. It also cannot be taken into account as an adequate evaluation of the code's usefulness and safety, its bug-free status, or any other contract clauses. Despite the fact that we did our best in conducting the analysis and putting together this report, it is crucial to note that you shouldn't rely solely on it. Instead, we advise moving forward with a number of independent audits and a public bug bounty program to ensure the security of smart contracts.

This audit does not constitute a formal partnership or relationship with the Company and should not be used to assume a business relationship.

Technical Disclaimer

On a blockchain network, smart contracts are set up and carried out. Hacking vulnerabilities may exist in the platform, the programming language used, and other applications used in conjunction with the smart contract. The explicit security of the audited smart contracts cannot therefore be guaranteed.