

GOLD PRLZ (gPRLZ) Contract Audit Report



Updated: January 06, 2026

Table of Contents

- Introduction
 - Disclaimer
 - Scope of Audit
 - Methodology
 - Security Review Summary
 - Security Analysis
 - Contract Structure
-

Introduction

The purpose of this report is to document the security analysis and contract structure of the `gPRLZ.sol` contract. This updated audit explores deeper into token mechanics, role interactions, and ERC20 compliance for enhanced insight.

Disclaimer

This report is based on the information provided at the time of the audit and does not guarantee the absence of future vulnerabilities. Subsequent security reviews and on-chain monitoring are strongly recommended.

Scope of Audit

The audit focused on the following aspects of `gPRLZ.sol`:

- Security mechanisms, including mint/burn controls and allowance handling
- Code correctness, logical flow, and standard compliance
- Adherence to best practices
- Gas efficiency and minimalism in design

Methodology

The audit process involved:

- Manual code review of inheritance and overrides
 - Automated analysis using Slither and Aderyn
 - Scenario-based testing using Foundry for mint/burn simulations
-

Security Review Summary

Review commit hash:

- `0c1fcf72ede03bf697c60f435e7a866f0312c9d3`

The `gPRLZ.sol` contract maintains core ERC20 security with role-based supply controls, ensuring controlled inflation/deflation.

Security Analysis

As an upgradeable ERC20 gPRLZ restricts supply changes to `MINTER_ROLE` (`mint`) and `BURNER_ROLE` (`burnFrom`), with `DEFAULT_ADMIN_ROLE` for role governance.

The `burnFrom` requires allowance (`_spendAllowance`), enabling delegated burns while preventing unauthorized destruction, Integrating with the `Cashier` contract's burn mechanics safely. No direct `burn` override exists, funneling all burns through roles for auditability.

Inheritance from OpenZeppelin ensures standard compliance, with `_mint` and `_burn` internals handling totalsupply/balance updates properly and safely.

Events (`gPRLZMinted`, `gPRLZBurned`) log changes for monitoring. Upgradeability (`Initializable`, 50-slot `__gap`) allow state variable extensions for future upgrades without data risks.

The design prioritizes simplicity and unauthorized supply alterations.

Contract Structure

The contract adheres to ERC20 standards with extensions:

- **State Variables:** Roles (`MINTER_ROLE`, `BURNER_ROLE`).
- **Initialization:** Sets name/symbol, initializes burnable/access, grants roles.
- **Core Functions:**
 - `mint`: Role-restricted supply increase with event.
 - `burnFrom`: Allowance-checked burn with event.