Genesis Keys (ACRE, PLOT, YARD) Contracts Audit Report

Updated: January 08, 2026

Table of Contents

Introduction

The purpose of this report is to document the security analysis and contract structure of the `ACRE`, `PLOT`, and `W` contracts, collectively referred to as Genesis Keys. These contracts share identical logic for minting, batch management, and access controls, providing entry points to the VMCC/COA ecosystem through NFT-based keys.

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 the Genesis Keys contracts (`ACRE.sol`, `PLOT.sol`, `YARD.sol`):

- Security mechanisms, including role-based access, blacklisting, batch controls, and payment integrations
- Code correctness, logical flow, and compliance with ERC721 standards
- Adherence to best practices for upgradeability, minting processes, and referral/fee splitting
- Gas efficiency, error handling, and prevention of unauthorized mints or transfers

Methodology

The audit process involved:

- Manual code review of inheritance, overrides, minting flows, and batch management
- Automated analysis using Slither and Aderyn to detect common vulnerabilities like access control issues or reentrancy
- Scenario-based testing using Foundry for simulating mints, batch updates, payments, referrals, and edge cases (e.g., max buy limits or blacklisted users)

Security Review Summary

Review commit hash:

- `6ca5938102158376980e9e5a1d5698a03f3e7e47`

The Genesis Keys contracts provide a secure NFT minting system with batch-based supply control and integrated payments/referrals, ensuring controlled distribution and ecosystem access.

Security Analysis

As upgradeable ERC721 contracts, Genesis Keys inherit from OpenZeppelin's ERC721Upgradeable and AccessControlUpgradeable, enabling safe initialization and role management. The SIGNER_ROLE restricts critical operations like batch setups, payment token configurations, blacklisting, and free minter assignments, while DEFAULT_ADMIN_ROLE handles initial grants and transfers.

Minting functions (`mint`, `mintAsFreeMinter`) enforce blacklisting, payment validations, batch availability, and quantity limits to prevent unauthorized or excessive mints. Payments use IERC20 transfers to a fee splitter, with optional referral processing if configured. Transfers override standard ERC721 methods to apply blacklisting, reducing risks from malicious actors.

Batch management ensures sequential supply distribution, with updates restricted to signers. Free minting is gated by participant controllers, adding flexibility without compromising security. URI is fixed per contract, simplifying metadata handling.

Events (e.g., NewBatchCreated, BatchUpdated) enable monitoring. No explicit reentrancy guards, but external calls (e.g., to fee splitter/referral) are post-state updates. Blacklisting prevents interactions from flagged addresses.

Automated tools flagged no high-severity issues, though minor gas optimizations (e.g., in loops) were noted. Testing confirmed robustness against over-minting, invalid payments, and unauthorized actions.

The contracts prioritize security through simplicity, explicit checks, and audited dependencies.

Contract Structure

The contracts extend upgradeable ERC721 with custom minting and batch logic (identical across ACRE, PLOT, YARD):

- **State Variables**: Batch structs, payment tokens, mappings for batches, tokens to batches, free participants/controllers, blacklists; constants for roles.
- **Initialization**: Grants admin role, sets ERC721 name/symbol, enables initial payment token.
- **Core Functions**:
  - `mint`: Handles paid minting with quantity checks, payments, fee splitting, and referrals.
  - `mintAsFreeMinter`: Free minting for authorized participants.
  - `setCurrentBatch` / `updateCurrentBatch`: Signer-restricted batch creation/updates.
  - `setPaymentToken`: Configures supported tokens and reward eligibility.
  - `setBaseURI` / `tokenURI`: Metadata management.
  - Transfer overrides: Apply blacklisting.
  - Admin setters: For max buy, free controllers/participants, referrer manager, fee splitter, blacklisting.
- **Helpers**: Internal payment processing, blacklisting/authorization checks.

- **Inter-Contract Integration**: Interfaces with IFeeSplitter and IReferralManager for fee/reward distribution.