Financing System 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 FinancingSystem and Mortgage contracts. This audit examines financing mechanics, mortgage management, interest calculations, payment processing, and asset transfers for a secure and efficient digital asset financing system.

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 FinancingSystem.sol and Mortgage.sol:

- Security mechanisms, including role-based access, blacklist checks, default detection, and credit management
- Code correctness, logical flow, and compliance with ERC20/ERC721 standards
- Adherence to best practices for upgradeability, interest calculations, and integration between contracts
- Gas efficiency, error handling, and prevention of unauthorized actions or financial exploits

Methodology

The audit process involved:

- Manual code review of inheritance, overrides, financing flows, and mortgage interactions
- Automated analysis using Slither and Aderyn to detect common vulnerabilities like arithmetic overflows or access control issues
- Scenario-based testing using Foundry for simulating financing applications, payments, defaults, foreclosures, and edge cases (e.g., full upfront payments or credit usage)

Security Review Summary

Review commit hash:

- `1fcf72ede03bf697c60f435e7a866f0312c9d3gf`

The FinancingSystem and Mortgage contracts provide a robust framework for asset financing and mortgage management, with integrated credit systems and role-restricted operations to prevent unauthorized modifications or exploits.

Security Analysis

As upgradeable contracts, `FinancingSystem` and `Mortgage` inherit from OpenZeppelin's `AccessControlUpgradeable` and `Initializable`, ensuring safe initialization and role-based governance. The `SIGNER_ROLE` in `FinancingSystem` restricts configurations like asset setups, payment tokens, and blacklisting, while `MORTGAGE_ROLE` in Mortgage controls critical actions like credit consumption and asset transfers.

Financing flows in `FinancingSystem` enforce mandatory credit usage when available, with validations for down payments, interest accruals, and defaults based on time thresholds. Payments use SafeERC20 for secure token transfers, and calculations (e.g., total cost, interest per second) avoid overflows through basis points and precision constants. Integration with Mortgage via role-restricted calls ensures atomic updates for mortgage creation, payments, and completions/foreclosures.

Mortgage handles NFT minting and transfers securely via IATL interface, tracking payments and statuses without direct user modifications. Credit management accumulates from payments and foreclosures, consumable only by authorized roles. Blacklisting prevents malicious users from initiating financings.

Events (e.g., FinancingCreated, MortgageCreated) provide auditability. Upgradeability is supported with storage gaps (98 in FinancingSystem, 100 in Mortgage) for future expansions. No reentrancy guards are explicit, but external calls are minimized and placed after state updates.

Automated tools flagged no high-severity issues, though minor optimizations (e.g., in mappings) were noted. Testing confirmed protection against overpayments, unauthorized transfers, and default manipulations.

The system prioritizes security through explicit validations, audited dependencies, and inter-contract trust minimization.

Contract Structure

The contracts extend upgradeable standards with custom financing and mortgage logic:

FinancingSystem

- **State Variables**: Asset configs, user financings, payment details, blacklists; mappings for user data; constants for precision.
- **Initialization**: Grants roles and initializes access control.
- **Core Functions**:
  - `applyForFinancing`: Initiates financing with down payment, credit usage, and mortgage creation.
  - `makeDownPayment`: Processes ongoing payments, updates states, and completes if fully paid.
  - `forfeitDefaultedAsset`: Handles asset foreclosure on defaults.
  - Configuration setters (e.g., `setAssetConfig`, `setPaymentTokenDetails`) for roles.
- **Getters**: Extensive views for user financing details, asset configs, and payment info.
- **Helpers**: Interest calculations, token transfers, default checks.

Mortgage

- **State Variables**: Mortgage data, payments, credit info; mappings for user mortgages and assets.
- **Initialization**: Grants roles and sets initial mortgage ID.
- **Core Functions**:
  - `createMortgage`: Sets up new mortgage with initial payment.
  - `makeDownPayment`: Records payment updates.
  - `transferAsset`: Mints and transfers NFT on completion.
  - `foreCloseAsset`: Adds credit on foreclosure.
  - `consumeCredit`: Deducts credit for financing.
- **Getters**: Mortgage details, payments, available credit.
- **Helpers**: Status updates, ERC721 receiver.
- **Inter-Contract Integration**: FinancingSystem calls Mortgage for mortgage ops; shared roles ensure controlled access.