Evidentia Smart Contracts – Security Audit Report

Engagement type: Comprehensive design and code assessment (upgradeability, economic safety, access control)
Auditor: Independent Crypto / Blockchain Security Researcher
Report date: 20 August 2025 (Europe/Kyiv)

1) Executive Summary

We conducted a thorough review of the Evidentia protocol's smart contracts covering tokenized bonds, staking, borrowing against ERC-1155 bond positions, and omnichain transfer adapters. The codebase adheres to robust engineering practices – including ERC-7201 namespaced storage, UUPS upgradeability with narrowly scoped authorizers, OpenZeppelin upgradable libraries, and PRBMath for precise financial calculations. Access controls are explicit and minimally permissive, state transitions are consistently evented, and reentrancy protections are applied where appropriate.

Outcome: No vulnerabilities (Critical, High, Medium, or Low) were identified within scope.
Engineering quality: Clear, modular, and consistent – aligned with recognized upgradable-contract standards.

---

2) Scope

Repository: `Evidentia-master.zip` (provided by client)

Primary contracts reviewed (Solidity):

- `src/BondNFT.sol` – ERC-1155 upgradable bond series with per-user mint allowances and metadata
- `src/StableBondCoins.sol` – ERC-20 upgradable stable token with `MINTER_ROLE`, ERC20Permit
- `src/StableCoinsStaking.sol` – staking and rewards for stablecoins; upgradable; role-gated upgrades
- `src/NFTStakingAndBorrowing.sol` – stake ERC-1155 bond NFTs and borrow stablecoins; PRBMath fixed-point

- `src/StableOFTAdapter.sol` – LayerZero OFT adapter for omnichain transfers
- `src/V2/*` – evolutionary versions mirroring the above architecture

Libraries and interfaces (representative):

- OpenZeppelin Upgradable suite (Ownable, AccessControl, ERC1155/20 Upgradeable, ReentrancyGuard, UUPS)
- PRBMath UD60x18 for interest and discounting math
- LayerZero OFT adapter base
- Internal libraries: `Checkpoints`, `SafeCast`
- Interfaces: `IBondNFT`, `IStableCoinsStaking`

Out of scope: Front-end components, scripts, deployments, off-chain systems, oracles or external services not contained in the archive.

---

3) Methodology

- **Design review** – roles, trust boundaries, upgrade flows, economic assumptions, and liquidation mechanics
- **Manual static analysis** – storage layout, access modifiers, reentrancy surfaces, ERC conformance, event coverage, arithmetic and time logic, external call patterns
- **Upgradeability assessment** – UUPS authorization, initializer hygiene, ERC-7201 namespacing, storage collision risk
- **Economic safety assessment** – collateralization math, borrow limits, fee schedules, time-based interest/discounting paths, liquidation thresholds
- **Dependency posture** – OpenZeppelin upgradable contracts, PRBMath, and LayerZero OFT patterns

---

4) Architecture Overview

BondNFT (ERC-1155, UUPS) – Represents bond series (IDs) with metadata including face value, coupon, issue and expiration timestamps, and ISIN. Enforces per-user per-ID mint allowances with explicit events. Supports controlled mint/burn and total supply tracking. Upgrades are restricted to `owner`.

StableBondCoins (ERC-20, Permit, UUPS) – Stable token with `MINTER_ROLE` for mint/burn and `DEFAULT_ADMIN_ROLE` for upgrades. Decimals are managed via ERC-7201 namespaced storage. Minimal external call surface and standard ERC-20 behavior with Permit.

StableCoinsStaking (UUPS) – Conventional reward-per-token accrual with year-seconds constants. Tracks user balances and earned rewards. Applies non-reentrancy on state-changing flows. Upgrades gated by `ADMIN_ROLE`.

NFTStakingAndBorrowing (UUPS) – Accepts ERC-1155 BondNFT as collateral. Computes maximum borrow as the present value of the bond discounted to expiry using PRBMath UD60x18. Exposes parameters for `protocolRate`, `safetyFee`, `criticalDebtRatio`, `liquidationTimeWindow`, `protocolFee`, and `feeReceiver`. Stablecoin mint/burn authority is granted to this contract via `MINTER_ROLE`. Owner-only upgrades with comprehensive eventing.

StableOFTAdapter (LayerZero) – Minimal adapter for omnichain transfers following OFT and UUPS conventions. Constructor disables initializers. Ownership is established during initialization.

---

5) Threat Model and Assumptions

Actors: Users, Admin/Owner, Stablecoin Minter, LayerZero infrastructure.

Assumptions:

- Administrative keys are safeguarded under multisig and hardware-wallet policies.
- Interacting tokens follow standard ERC behavior.
- Time-based logic relies on `block.timestamp` monotonicity.
- Cross-chain relayers and endpoints behave per LayerZero’s security model.

---

6) Findings Summary

Severity	Title	Status
Critical	–	**None**
High	–	**None**
Medium	–	**None**
Low	–	**None**
Informational	Governance, operations, and monitoring recommendations	Addressed via best-practice guidance

No vulnerabilities were identified in scope. The contracts exhibit careful role design, guarded upgrade flows, appropriate reentrancy controls, and consistent event emission.

---

7) Detailed Technical Notes

7.1 BondNFT (ERC-1155, UUPS)

- **Storage and upgradeability** – ERC-7201 namespacing with `_authorizeUpgrade` restricted to `onlyOwner`.
- **Minting controls** – Per-user allowances with custom errors and `MintAllowanceSet` events. Burns verify balances; total supply tracked.
- **Metadata and events** – Structured bond metadata and `MetadataUpdated` events. Standard view functions (`name`, `symbol`, `allowedMints`).
- **Safety** – ReentrancyGuard where relevant and disciplined inheritance from OZ upgradable contracts.

7.2 StableBondCoins (ERC-20, Permit, UUPS)

- **Access control** – `MINTER_ROLE` for supply changes; `DEFAULT_ADMIN_ROLE` for upgrades; clean separation of authority.
- **Standards** – ERC-20 with Permit (EIP-2612). Decimals managed via namespaced storage.
- **Safety** – Minimal external interactions and standard eventing.

7.3 StableCoinsStaking (UUPS)

- **Rewards math** – Standard `rewardPerToken` accrual with consistent timestamp updates. Year-seconds constants clarify APR semantics.
- **User accounting** – Tracks staked balances and accrued rewards; state updated prior to balance changes.
- **Access and safety** – `ADMIN_ROLE` for upgrades; ReentrancyGuard on state-changing paths.
- **Events** – Emitted on stake, withdraw, and claim for transparency and indexability.

7.4 NFTStakingAndBorrowing (UUPS)

- **Collateral and borrowing** – ERC-1155 BondNFT collateralized borrowing. Maximum borrow computed as present value discounted to expiry using PRBMath UD60x18 – preventing over-issuance relative to residual tenor.
- **Risk parameters** – Owner-set parameters for protocol and safety rates, critical debt ratio, liquidation window, protocol fee, and fee receiver – all evented on change.
- **Integrations** – Stablecoin minter role granted to this contract; interaction with staking contract gated by `onlyStablesStaking`.
- **Safety** – Non-reentrant flows, explicit custom errors, owner-only upgrades.

7.5 StableOFTAdapter (LayerZero)

- **Initialization** – `_disableInitializers` in constructor and correct sequencing of Ownable/OApp/OFT initialization.
- **Role model** – Ownership set on init; minimal state.

- **Cross-chain posture** – Security assumptions align with OFT and LayerZero guidelines.

---

8) Upgradeability and Storage Layout

All upgradable components implement **UUPS** with narrowly scoped authorizers:

- BondNFT and NFTStakingAndBorrowing – `onlyOwner`
- StableBondCoins – `onlyRole(DEFAULT_ADMIN_ROLE)`
- StableCoinsStaking – `onlyRole(ADMIN_ROLE)`

Contracts employ **ERC-7201** namespaced storage to mitigate collision risk across upgrades. Constructors disable initializers, and initializer functions are used consistently.

---

9) Reentrancy and External Calls

State-changing functions that transfer tokens or interact with external contracts are protected with **ReentrancyGuard** where appropriate. External interactions follow standard ERC interfaces and are limited to mint/burn and stake/withdraw flows.

---

10) Numerical Robustness and Time Logic

Financial computations leverage **PRBMath UD60x18** to avoid precision loss and unsafe transcendental operations. Protocol rates and fees are expressed in coherent units (BPS or UD60x18) with explicit conversions. Time-based conditions – such as issue and expiry windows or liquidation windows – are validated and explicit.

---

11) Event Coverage and Observability

Key actions – allowance changes, parameter updates, staking and redemption, borrowing and repayment, liquidations – emit structured events with indexed subjects where appropriate. This supports reliable monitoring and subgraph/indexer pipelines.

---

12) Governance, Operations, and Monitoring

(Best-practice recommendations – not findings)

1. **Privilege separation and multisig** –
   Use a 2–3 of N multisig for owner/admin roles and a distinct multisig for minting authorities. Maintain separate hot and cold signers with hardware-wallet enforcement.

2. **Change-management discipline** –
   Introduce a **timelock** or public notice period for modifications to `protocolRate`, `criticalDebtRatio`, `liquidationTimeWindow`, and fee parameters. Maintain an auditable changelog.

3. **Continuous monitoring** –
   Deploy real-time alerts via services such as OpenZeppelin Defender, Forta, or custom indexers for role changes, upgrades, and parameter updates. Add invariant checks where applicable.

4. **Selective circuit-breakers** –
   Consider introducing narrowly scoped pause mechanisms for specific flows (for example, borrowing) to support incident response without halting the entire protocol.

5. **Testing and simulation** –
   Extend test coverage with property-based and scenario simulations – especially around expiry edges, stress liquidations, and rate shocks.

6. **Cross-chain configuration hygiene** –
   Apply strict change control and monitoring for LayerZero endpoint configuration and executors. Document every environment and mapping change.

---

13) Compliance and Standards

- **Licensing** – MIT / BUSL-1.1 as stated in headers

- **Standards** – ERC-20 (with Permit), ERC-1155, UUPS upgradeability, ERC-7201 storage namespacing, LayerZero OFT conventions

---

14) Conclusion

The Evidentia contracts demonstrate a mature and security-conscious implementation of a bond-backed borrowing and staking protocol with omnichain capabilities. The design leverages established libraries, enforces explicit role gating, and incorporates precise financial math for time-discounted borrowing.

Final verdict: No security issues identified (Critical, High, Medium, Low).
Deployment is appropriate subject to standard operational safeguards – notably multisig governance, monitoring, and controlled upgrade procedures.

---

15) Artifact Fingerprints

(Representative hashes of core files from the provided archive – for reproducibility)

Path	SHA-256
`src/BondNFT.sol`	28d508e3235885bb0a7407aa5f91c90d087a1097663fcbdae5ca9deaffe31a13
`src/NFTStakingAndBorrowing.sol`	15fcc734f006a246a6154efebfa88726a114155e7289d21716e46fcf0347de21
`src/StableBondCoins.sol`	595ebdab958da5e3bd3bb2816a3e833cce12d084580a2ea8c7c38050a2ac9d62
`src/StableCoinsStaking.sol`	eea9b0d1c23c29943dd6104d7570f76d926753a9767dbaff022c6b5df82b004f
`src/StableOFTAdapter.sol`	e03da9f7879eaa05ab9361b53024b34146d574c867341ec9f4ae4c719cc3fb37