DEV SCAN

Pre-Launch Readiness Check

Automated testnet deployment review — technical transparency, role hygiene, and evidence readiness before mainnet.

Project	**Riverside Apartments**
Network	**base_sepolia**
Environment	**Testnet**
Scan type	**Automated · no human audit**
Date	**2026-05-14**

EXECUTIVE ASSESSMENT

Ready for external review

The scan resolves a nine-contract UUPS upgradeable system on base_sepolia testnet — one primary token plus eight related contracts (sale, settlement, vesting, registry, valuation, distribution, staking, referral) — and finds all nine source-verified, all nine implementation contracts cleared by the configured static-analysis suite (0 findings across the protocol), and seven AccessControl roles defined on the primary with the deployer EOA holding six of them. The pre-launch readiness gap is operational rather than code-level: privileged custody is concentrated on a single externally-owned account, none of the two probed administrative addresses match Gnosis Safe bytecode, and the testnet profile means holder distribution, DEX liquidity, and fund-flow attribution become assessable only after mainnet deployment.

Testnet interpretation — missing multisig is not automatically a blocker on testnet; missing role separation and unverified source are weak readiness signals before mainnet.

9 / 9 contracts verified

EOA-controlled roles

Proxy visibility role-gated

SCAN VERDICT

SUBJECT

HIGH

ANALYSIS

PARTIAL

7 of 11 modules applicable

Source visibility	**PASS**
Role separation	**PASS**
Upgrade authority separation	**PASS**
Access control pattern	**PASS**

Static analysis: Slither + Mythril + Aderyn + Semgrep, 0 findings (implementation)

TRANSPARENCY

Source verified

9 of 9 contract(s) verified on the testnet explorer. Code-level review is feasible from public evidence.

READINESS GAP

Privileged roles concentrated on the deployer EOA

The deployer holds **6 privileged role(s)** across the in-scope contracts. Plan role-transfer and multisig wrapping before mainnet.

OUT OF SCOPE ON TESTNET

No production market signals

Holder concentration, DEX liquidity depth, LP ownership, and real fund-flow provenance require a mainnet scan. Plan a re-run after the mainnet deployment.

Observed result vs. expected testnet readiness

AREA	OBSERVED RESULT	EXPECTED TESTNET RESULT	REQUIRED BEFORE MAINNET	STATUS
Source verification	9 of 9 contracts verified	Primary implementations verified	Verify remaining contracts before mainnet	OK
Privileged control	Roles held by deployer EOA (6 role(s) on the deployer key)	Role separation visible, even if multisig is deferred	Document production owner / admin / role structure	CONCENTRATED
Multisig custody	0 of 2 probed address(es) backed by a multisig	Optional on testnet	Use multisig / timelock — or justify mainnet custody model	NOT BLOCKING ON TESTNET
Upgrade authority	UUPS proxy; 9 of 9 contracts pass; UPGRADER_ROLE separated from deployer on all	UPGRADER_ROLE on multisig (Safe) is recommended pre-mainnet	Optional: migrate UPGRADER_ROLE to a Safe	OK

Important limitation — automated public-evidence scan. Not investment, financial, legal, tax, regulatory, MiCA, or audit advice.

Use as — pre-launch readiness input · not a production risk conclusion.

Verdict axes (authoritative basis)

The dashboard above summarises the report. The block below is the scan's ASVS-style verdict that the dashboard derives from: each transparency axis with its outcome, modality, and quantitative basis. If any dashboard chip appears to disagree with an axis outcome here, the axis outcome below is authoritative; the dashboard chip is a heuristic restatement.

Pre-launch transparency (testnet) · **Verdict logic v6**

Subject verdict: HIGH
Analysis verdict: PARTIAL (7 of 11 modules applicable)

Axes (ASVS-style):

Axis	Modality	Outcome	Detail
source visibility	MUST	**PASS**	
role separation	SHOULD	**PASS**	distinct_addresses = 2
upgrade authority separation	SHOULD	**PASS**	proxy_pattern = uups, role_label = UPGRADER_ROLE, role_holders_count = 1, upgrader_separated = 1, contracts_checked = 9, contracts_passing = 9, contracts_failing = 0, contracts_skipped_n_a = 0, contracts_skipped_data_missing = 0, failing_addresses = [], admin_backdoor = 1, all_safes = 0
access control pattern	SHOULD	**PASS**	

Pre-launch profile — three transparency axes (holder distribution, DEX liquidity, fund-flow attribution) become assessable only after mainnet deployment and are excluded from the scan verdict for this profile. Source verification is supported on testnet explorers but is not universally adopted; whether to verify on testnet is an operational choice rather than a default expectation.

2. Modules examined

The framework checks eleven modules against the scan's evidence. The two tables below summarise the result for each module and the privileged accounts surfaced across modules; the per-module subsections that follow go into detail.

Module results — at a glance

Module	Status	Result at a glance
Contract inventory	Examined	1 primary + 8 related contract(s) in scope
Source code verification	Examined	9 of 9 verified on the explorer
Proxy and upgradeability	Examined	yes (uups)
Ownership and admin	Examined	no Ownable owner observed
Access control roles	Examined	yes (7 role(s))
Deployer analysis	Examined	EOA; holds 6 privileged role(s)
Multisig wallet detection	Examined	0 of 2 privileged address(es) backed by a multisig
Holder distribution	Not relevant on testnet	not assessed (testnet profile)
DEX liquidity	Not relevant on testnet	not assessed (testnet profile)
Fund flow	Not relevant on testnet	not assessed (testnet profile)
Static analysis	Examined	Slither + Mythril + Aderyn + Semgrep ran on implementation; 0 findings

Privileged accounts

Address	Type	Privileges held	Source
`0x1de7378031695df3c920d0a167390f84bed57447`	EOA (not a Safe)	`BURNER_ROLE`, `DEFAULT_ADMIN_ROLE`, `PAUSER_ROLE`, `custom role (0x224b562a…)`, `custom role (0x442a94f1…)`, `custom role (0x5fdbd35e…)`, `deployer`	access_control, deployer_analysis, safe_detection
`0xfbe3b46519fc1f9ca1d6279018b713d5b2f8b499`	Not assessed (not a Safe)	`UPGRADER_ROLE`	access_control, safe_detection

Type column legend. `EOA` = externally-owned account; `contract` = smart contract (not classified further); `multisig (Safe, M/N)` = identified Gnosis Safe with M-of-N threshold; `X (not a Safe)` = contract probed but did not match Safe bytecode; `Not assessed` = type not classified.
The scan probed 2 privileged contract address(es) for Gnosis Safe bytecode patterns; no Safes were identified, so those addresses are annotated `(not a Safe)` in the Type column. Timelock detection is not implemented in the current scanner.

Per-module detail

Each module subsection has the same shape: what it checks (Purpose), what was found in this scan (deterministic), why it matters in general plus scan-specific elaboration, and what the result means for this scan specifically.

2.1 Contract inventory

Purpose — Lists every contract addressed by this scan: the primary token contract and any related contracts (sale, vesting, treasury) discovered during evidence collection.

What was found — Status: *Examined*

- Primary contract: `0x33C2524bd6BC3278292B188F731541FD1D22e572`
- Related contracts in scope: **8**

Role	Address	Source verification
sale	`0x43d180E3e4DF999438aEE63B7f57f162F7E4Ebd7`	verified
settlement	`0xAACd34940A7df20fE5953d2432a5084e84C4A375`	verified
vesting	`0x818ccA523d79c5fd21B62D91646cd1773a2C7c59`	verified
registry	`0xb0e6c53Bb4CF047BDB1ad35fB2707A627a01FaD7`	verified
valuation	`0x787f11306D1dF3785cC3Fe131A47737D14D92F04`	verified
distribution	`0x98315B281992097afc0c3A44b09e8Ea8bB5f3168`	verified
staking	`0x377fA3430F1f6154EA5AFBc577fDdF67a5a9bA81`	verified
referral	`0x9ef078ee22ad58bcae3f6a2e423e0804a13ed699`	verified

Why it matters — A reader needs to know which addresses are in scope before drawing any conclusion. A token contract that depends on related sale or vesting contracts must be evaluated as a system.

The Riverside Apartments deployment is a multi-contract protocol: the `RealEstateToken` primary plus eight related contracts spanning sale, settlement, vesting, registry, valuation, distribution, staking, and referral — every related contract must be evaluated as part of a connected system rather than in isolation.

What it means —

The nine-contract layout enumerated above is the unit of analysis for the rest of this report. Each related contract has its own role (e.g., `sale` orchestrates primary issuance, `staking` mediates locked positions) and its own implementation behind a UUPS or EIP-1967 proxy (see §2.3). Because the contracts interact (the token grants role-bearing access to settlement, vesting reads token balances, etc.), an evaluation has to account for cross-contract privilege flow rather than treating each contract as standalone. The cross-contract AccessControl probe in §2.5 and the cross-contract drill-through static-analysis result in §2.11 both report on the eight related contracts in addition to the primary, but per-related role enumeration is deferred at this scanner version.

2.2 Source code verification

Purpose — Checks whether the contracts' source code is publicly verified on the blockchain explorer. Verified source means anyone can read and audit the actual logic; unverified means only the compiled bytecode is visible.

What was found — Status: *Examined*

- Verified on the explorer: **9** contract(s) — `0x33C2524bd6BC3278292B188F731541FD1D22e572`, `0x43d180E3e4DF999438aEE63B7f57f162F7E4Ebd7`, `0xAACd34940A7df20fE5953d2432a5084e84C4A375`, `0x818ccA523d79c5fd21B62D91646cd1773a2C7c59`, `0xb0e6c53Bb4CF047BDB1ad35fB2707A627a01FaD7`, `0x787f11306D1dF3785cC3Fe131A47737D14D92F04`, `0x98315B281992097afc0c3A44b09e8Ea8bB5f3168`, `0x377fA3430F1f6154EA5AFBc577fDdF67a5a9bA81`, `0x9ef078ee22ad58bcae3f6a2e423e0804a13ed699`
- Unverified: **0** contract(s)
- Implementation contract `0x12a1854d2e0c65b78098963af3d91b46067ca617` source is **verified** on the explorer (source `RealEstateToken.sol`, compiler `v0.8.33+commit.64118f21`). Code-level analysis of the executing logic is feasible.

Why it matters — Verified source is the precondition for every other code-level check, including static analysis and independent audit. Without it, no one outside the development team can read what the contract actually does.

All nine in-scope proxy contracts and the `RealEstateToken` implementation behind the primary are verified on the testnet explorer — the entire on-chain code surface (proxy shells + implementations) is publicly readable.

What it means —

For a testnet deployment, whole-protocol source verification is a strong pre-launch readiness signal. Implementation drill-through (see §2.3) extends the verification coverage to the actual logic contracts behind each proxy, so a code reviewer can read every contract that will execute. The static-analysis result in §2.11 operated on this same source set across all nine implementations and is therefore as comprehensive as the rule libraries allow. Source verification is a precondition for code review, not a substitute for it — a manual review or Tier 2 audit remains the next step.

2.3 Proxy and upgradeability

Purpose — Determines whether the primary contract is a proxy that delegates to a separate implementation contract, and if so, who controls upgrades.

What was found — Status: *Examined*

- Is a proxy: **yes** (pattern: `uups`)
- Implementation address: `0x12a1854d2e0c65b78098963af3d91b46067ca617`
- Upgrade administrator: *not provided*

Proxy / implementation map (per-contract drill-through):

Role	Proxy address	Implementation address	Implementation contract	Status
primary (token)	`0x33C2524bd6BC3278292B188F731541FD1D22e572`	`0x12a1854d2e0c65b78098963af3d91b46067ca617`	`RealEstateToken` (uups)	source `verified`, analysis `checked`
sale	`0x43d180E3e4DF999438aEE63B7f57f162F7E4Ebd7`	`0x5f0760c461c8ee2c0e57f774e954c0f40ed7a762`	`RealEstateSale` (EIP-1967)	source `verified`, analysis `checked`
settlement	`0xAACd34940A7df20fE5953d2432a5084e84C4A375`	`0x8d9b67f07f3ba9493d3b62cfa3aab2aa49f593cf`	`RealEstateEscrowSettlement` (EIP-1967)	source `verified`, analysis `checked`
vesting	`0x818ccA523d79c5fd21B62D91646cd1773a2C7c59`	`0x06ba7a25b1008ab6f3002e6efeb2e8167e661268`	`RealEstateVesting` (EIP-1967)	source `verified`, analysis `checked`
registry	`0xb0e6c53Bb4CF047BDB1ad35fB2707A627a01FaD7`	`0x74dabd75b0aba74eb42127cb6a531b4326d6addf`	`RealEstateAssetRegistry` (EIP-1967)	source `verified`, analysis `checked`
valuation	`0x787f11306D1dF3785cC3Fe131A47737D14D92F04`	`0x61da1a4dc055806a0149ff98ef90782d7da708cc`	`RealEstateAssetValuation` (EIP-1967)	source `verified`, analysis `checked`
distribution	`0x98315B281992097afc0c3A44b09e8Ea8bB5f3168`	`0x4f2ea3505660eab333368a4b8e641249fc1e9e7d`	`RealEstateRevenueDistribution` (EIP-1967)	source `verified`, analysis `checked`
staking	`0x377fA3430F1f6154EA5AFBc577fDdF67a5a9bA81`	`0x6aca2db354c9515d1805371220465d28eeb17ee6`	`RealEstateStaking` (EIP-1967)	source `verified`, analysis `checked`
referral	`0x9ef078ee22ad58bcae3f6a2e423e0804a13ed699`	`0x70c80dbbf8fa3ab692d5d3f33ff059fdfd575569`	`ReferralRewardDistributor` (EIP-1967)	source `verified`, analysis `checked`

Why it matters — An upgradeable contract can change behaviour after deployment. The address (or governance) controlling the upgrade is therefore as important as the current code.

The primary is a UUPS upgradeable proxy delegating to `RealEstateToken`; the eight related contracts each follow the EIP-1967 storage layout with a separate verified implementation behind each proxy — upgradeability is the structural mechanism for the whole protocol, not a defect.

What it means —

UUPS means the upgrade logic lives in the implementation contract (`RealEstateToken`) rather than in the proxy itself; whoever holds the `UPGRADER_ROLE` on the implementation can authorise an `upgradeToAndCall` and change the executing logic. The pre-launch readiness chart in §1 records that `UPGRADER_ROLE` is separated from the deployer on all nine contracts — a meaningful role hygiene signal even though both `UPGRADER_ROLE` and the deployer's other roles still resolve to externally-owned accounts rather than multisigs. Migrating `UPGRADER_ROLE` to a Gnosis Safe before mainnet is the operational hardening step explicit in the pre-launch readiness table.

2.4 Ownership and admin

Purpose — Identifies the contract owner (the address with elevated administrative power under the Ownable pattern) and whether ownership has been renounced.

What was found — Status: *Examined*

- Owner: **none observed (ownership renounced or not Ownable)**

Why it matters — The owner address holds privileged powers defined by the contract (e.g., minting, transfer pause, ownership transfer). Ownership status directly bounds operational risk.

No Ownable owner is recoverable from the standard `owner()` probe — this protocol uses OpenZeppelin AccessControl (see §2.5) rather than the Ownable single-owner pattern.

What it means —

The absence of an Ownable owner is not a coverage gap; it reflects the deliberate design choice to use role-based access control instead. The seven roles enumerated in §2.5 (`DEFAULT_ADMIN_ROLE` , `UPGRADER_ROLE` , `BURNER_ROLE` , `PAUSER_ROLE` , plus three custom-hash roles) are the actual privilege surface, and the privileged-accounts table at the top of §2 reflects who currently holds them. Reviewing the Ownable probe alone would understate the privilege architecture; the AccessControl module is the right instrument for this protocol.

2.5 Access control roles

Purpose — Inspects role-based permissions defined under the OpenZeppelin AccessControl pattern: which roles exist and which addresses hold each role. Roles typically govern minting, burning, pausing, and upgrades.

What was found — Status: *Examined*

- AccessControl detected: **yes**
- Distinct roles defined: **7** — `DEFAULT_ADMIN_ROLE` , `UPGRADER_ROLE` , `custom role (0x224b562a…)` , `BURNER_ROLE` , `custom role (0x442a94f1…)` , `custom role (0x5fdbd35e…)` , `PAUSER_ROLE`
- Unique addresses across all roles: **2**
- Cross-contract AccessControl probe: **8 of 8** related contracts expose AccessControl (`sale` , `settlement` , `vesting` , `registry` , `valuation` , `distribution` , `staking` , `referral`). Per-related role enumeration is deferred and not in scope for this scan.

Why it matters — Role-based access control distributes contract powers across multiple addresses. Concentration of multiple sensitive roles in one address reduces the operational checks the contract was designed to provide.

The primary contract defines seven AccessControl roles — `DEFAULT_ADMIN_ROLE`, `UPGRADER_ROLE`, `BURNER_ROLE`, `PAUSER_ROLE`, plus three custom-hash roles — distributed across only two unique addresses, and all eight related contracts also expose AccessControl (per-related role enumeration is deferred at this scanner version).

What it means —

Two unique addresses across seven roles means the operational reality is closer to a 2-key model than a 7-way role separation; one of those addresses is the deployer EOA holding six of the role hashes (see §2.6). The role-segmentation pattern that AccessControl provides is fully wired into the source, but the address-level distribution does not yet reflect it. Per-related-contract role enumeration is out of scope at this scanner version — the cross-contract AccessControl probe confirms that all eight related contracts expose the same `hasRole` / `getRoleAdmin` surface, but the specific role-holder mapping per related contract is not in this scan's evidence. Documenting the intended production owner / admin / role structure across the protocol before mainnet is the readiness gap surfaced in §1.

2.6 Deployer analysis

Purpose — Identifies the address that deployed the primary contract and whether the deployer retained any administrative privileges after deployment.

What was found — Status: *Examined*

- Deployer address: `0x1de7378031695df3c920d0a167390f84bed57447` (EOA)
- Deployer transaction count: 65
- Deployer holds privileged roles: **6** —
  `role:0x0000000000000000000000000000000000000000000000000000000000000000`,
  `role:0x224b562a599bb6f57441f98a50de513dff0de3d9b620f342c27a4e4a898ce8e2`,
  `role:0x3c11d16cbaffd01df69ce1c404f6340ee057498f5f00246190ea54220576a848`,
  `role:0x442a94f1a1fac79af32856af2a64f63648cfa2ef3b98610a5bb7cbec4cee6985`,
  `role:0x5fdbd35e8da83ee755d5e62a539e5ed7f47126abede0b8b10f9ea43dc6eed07f`,
  `role:0x65d7a28e3265b37a6474929f336521b332c1681b933f6cb9f3376673440d862a`

Why it matters — The deployer often becomes the initial admin. If the deployer also holds privileged roles after deployment, the launching team retains single-point operational control.

The deployer at `0x1de7…7447` is an externally-owned account holding six privileged role hashes on the primary — `DEFAULT_ADMIN_ROLE` plus five custom-hash roles — concentrating the protocol's administrative surface on a single key in the current testnet state.

What it means —

`DEFAULT_ADMIN_ROLE` is the AccessControl role-management role: a holder can grant and revoke any other role, including `UPGRADER_ROLE` and `BURNER_ROLE`. With the deployer EOA holding `DEFAULT_ADMIN_ROLE` on the primary (and the same address concentrating five other role hashes), the testnet privilege model is effectively single-key — appropriate for a pre-launch development environment

but not for production. The pre-launch readiness chart in §1 calls out “Plan role-transfer and multisig wrapping before mainnet” as the operational step before a mainnet re-scan; documenting which production address will hold each role is the further-diligence condition.

2.7 Multisig wallet detection

Purpose — Probes every privileged address (deployer, owner, upgrade administrator, AccessControl role members) for the bytecode of recognised multisig wallet contracts — primarily Gnosis Safe — and reports which administrative keys are multisig-protected versus single-signer EOAs or other contracts.

What was found — Status: *Examined*

- Privileged addresses probed: **2**
- Multisig wallets confirmed: **0**
- Probed but not identified as multisig: **2**

Why it matters — A multisig wallet (e.g., a Gnosis Safe configured with an M-of-N threshold) requires multiple co-signers to authorise any administrative action; this provides operational resilience against the loss or compromise of a single key. Privileged addresses that are EOAs or non-multisig contracts can act under a single-signature path.

Two unique privileged addresses were probed against Gnosis Safe bytecode patterns; neither matched, so both are currently externally-owned accounts rather than multisig wallets.

What it means —

On testnet, missing multisig coverage is not blocking — the testnet framing in §1 explicitly records “Optional on testnet”. The pre-launch operational task is migrating at least `UPGRADER_ROLE` and `DEFAULT_ADMIN_ROLE` to a Gnosis Safe (or justified custom administrator) before the mainnet deployment. The scan reports the current bytecode classification of the two probed addresses; the production custody decision lives in project planning rather than in this scan’s evidence boundary.

2.8 Holder distribution

Purpose — Measures how the token’s supply is spread across addresses. Highly concentrated supply means a small number of wallets can move the market or dominate governance.

What was found — Status: *Not relevant on testnet*

- Holders observed: **0**
- Total supply (raw units): *n/a*
- Top 1 holder: *n/a* of supply
- Top 10 holders: *n/a* of supply
- Top 100 holders: *n/a* of supply

Why it matters — High supply concentration creates market and governance risk. A single large holder can move price unilaterally; a small group can dominate any token-weighted vote.

Holder distribution is not assessed on this testnet scan because token balances on a testnet do not map to operational holder identity.

What it means —

The module status `not_applicable_testnet` reflects scan-profile semantics: testnet addresses are typically faucet-funded deployment accounts, not population-scale holders, and any concentration metric computed on them would not predict mainnet distribution. A mainnet re-scan after the production deployment is the path to assessing actual holder distribution — at that point the same module will compute top-N concentration and HHI against the live population.

2.9 DEX liquidity

Purpose — Searches major decentralized exchanges for liquidity pools that pair the token against well-known assets, and reports the reserves observed.

What was found — Status: *Not relevant on testnet*

- Not assessed under testnet profile — DEX liquidity becomes relevant only after mainnet deployment.

Why it matters — Visible DEX liquidity is one objective signal that the token can be exchanged today. Absence of liquidity does not mean a token is non-functional, but it removes one transparency anchor.

DEX liquidity is not assessed on testnet because production trading venues do not operate on test networks.

What it means —

The module status `not_applicable_testnet` reflects that the Uniswap and other AMM probe targets the scan uses for liquidity discovery exist only on production chains. Testnet pools, where they exist, are not market-functional liquidity. Liquidity assessment becomes a viable question after the protocol is deployed to its target mainnet, at which point the same module will report pool counts and observed reserves against real venues.

2.10 Fund flow

Purpose — Traces the addresses that funded the primary contract or the deployer one hop back, helping identify the operational origin of the deployment.

What was found — Status: *Not relevant on testnet*

- Funders observed at 1-hop: **0**

Why it matters — Funder identity at one hop is a transparency anchor. Funding from a well-known exchange or a public team address contextualizes deployment operations differently from funding routed through privacy mixers.

Fund-flow attribution is not assessed on testnet because the deployer's testnet funding history reflects faucet provisioning rather than operational counterparty relationships.

What it means —

The module status `not_applicable_testnet` reflects that one-hop funder identity on a test network does not carry the same meaning it does on mainnet: testnet ether is faucet-issued, not exchanged for value, so the funder graph is operationally uninformative. A mainnet re-scan after the production deployment is the path to fund-flow attribution; at that point the same module will trace deployer funders against labelled counterparties.

2.11 Static analysis

Purpose — Runs the configured suite of automated source-code analyzers against the verified contract source to flag known vulnerability patterns. The exact tool set is enumerated in the §2.11 detail bullets; findings of zero do not prove the code is safe — only that the patterns those tools recognize did not match.

What was found — Status: *Examined*

- Tools attempted on implementation: `slither` (0.11.5, status `completed`); `slither-check-upgradeability` (None, status `completed`); `slither-check-erc` (None, status `skipped`); `mythril` (Mythril version v0.24.8, status `completed`); `aderyn` (aderyn 0.6.8, status `completed`); `semgrep` (1.163.0, status `completed`)
- Findings reported on implementation: **0**
- Implementation under analysis: `0x12a1854d2e0c65b78098963af3d91b46067ca617` (source `RealEstateToken.sol`, compiler `v0.8.33+commit.64118f21`)
- Proxy-surface analysis was skipped (`proxy_stub_only_skipped`) — analysing the delegatecall stub adds no signal; the row above reflects the implementation contract that actually executes.
- **Cross-contract drill-through.** Static analysis ran on **9 of 9 implementations** (1 primary + 8 related); **0 findings** reported across the protocol.

Why it matters — Automated analyzers catch a fixed set of known anti-patterns quickly. Their value in this report is binary: either they ran and produced findings (then the findings are listed), or they could not run and the report explains why.

Six analyzers ran end-to-end against all nine implementation contracts (primary plus eight related) and reported zero findings across the protocol; the compiler version (Solidity 0.8.33) is recent enough that the scanner's known-bug database does not flag any matches.

What it means —

Whole-protocol cross-contract static-analysis coverage at zero findings is a stronger signal than a single-contract clean run: the rule libraries that ship with slither, slither-check-upgradeability, mythril, aderyn, and semgrep — including the tokenomics rule library for mint/cap/fee/blacklist/pause patterns — did not match anything across 1 + 8 implementations. That is the strongest pre-launch readiness signal in this scan, even though it is not equivalent to a manual code review or a Tier 2 audit. The slither-check-erc skip is expected because the implementation contracts are application-domain logic (real estate vesting, settlement, etc.) rather than generic ERC profiles.

3. Limitations and further-diligence conditions

3.1 Known limitations (framework summary)

The framework lists every limitation already recorded by the scan, plus every module whose coverage status is anything other than `checked`. The LLM-written prose below (§3.2) elaborates on these and translates each `further_diligence_condition` into plain English.

Module	Source / code	Further-diligence condition
engine	`static_analysis_proxy_stub_only_skipped`	Once implementation drill-through is enabled, re-scan to analyze the implementation contract.
engine	`testnet_only`	Production assessment requires mainnet re-scan.
holder_distribution	*status:* `not_applicable_testnet`	Testnet scan profile: Holder distribution evidence is not applicable until a mainnet re-scan is available.
dex_liquidity	*status:* `not_applicable_testnet`	Testnet scan profile: DEX liquidity evidence is not applicable until a mainnet re-scan is available.
fund_flow	*status:* `not_applicable_testnet`	Testnet scan profile: Fund flow evidence is not applicable until a mainnet re-scan is available.

3.2 Plain-language elaboration

This scan does not answer whether the token is safe at production scale; it answers whether the pre-launch surface is consistent with what the team described.

Five conditions apply.

First, the entire scan operates under the `testnet_only` profile (limitation code `testnet_only`). Three transparency axes — holder distribution, DEX liquidity, and fund-flow attribution — become assessable only after mainnet deployment and are excluded from the verdict block for this profile. Production assessment requires a mainnet re-scan.

Second, `static_analysis_proxy_stub_only_skipped`. Static analysis was not run against the proxy bytecode at any of the nine proxy addresses because proxies are delegatecall plumbing without business logic. The implementation drill-through path is enabled and ran against all nine implementations (the primary `RealEstateToken` plus eight related implementation contracts) — the §2.11 zero-findings result reflects analysis on the code that actually executes.

Third, `not_applicable_testnet` for `holder_distribution`. Token balances on testnet do not represent operational holders; the module records the status and skips the concentration calculation. A mainnet re-scan after the production deployment is the path to a population-level distribution.

Fourth, `not_applicable_testnet` for `dex_liquidity`. Production DEX venues do not run on testnets, so the liquidity probe targets are not present. Liquidity assessment becomes a viable question after the mainnet deployment.

Fifth, `not_applicable_testnet` for `fund_flow`. One-hop funder attribution on testnet reflects faucet provisioning rather than counterparty relationships and would mislead if treated as deployer-provenance evidence. A mainnet re-scan after the production deployment is the path to fund-flow attribution.

In addition to the framework-recorded items, the pre-launch readiness chart in §1 identifies an operational condition not strictly tracked under `limitations[]`: privileged role concentration on the deployer EOA (six role hashes including `DEFAULT_ADMIN_ROLE`) and the absence of multisig coverage on either probed address. Migrating at least `UPGRADER_ROLE` and `DEFAULT_ADMIN_ROLE` to a Gnosis Safe or equivalent administrator, and documenting the intended production owner / admin / role structure across all nine contracts, is the operational hardening step before a mainnet re-scan.

Appendix A: Contracts in scope

Address	Role	Source verification
0x33C2524bd6BC3278292B188F731541FD1D22e572	token (primary)	verified
0x43d180E3e4DF999438aEE63B7f57f162F7E4Ebd7	sale	verified
0xAACd34940A7df20fE5953d2432a5084e84C4A375	settlement	verified
0x818ccA523d79c5fd21B62D91646cd1773a2C7c59	vesting	verified
0xb0e6c53Bb4CF047BDB1ad35fB2707A627a01FaD7	registry	verified
0x787f11306D1dF3785cC3Fe131A47737D14D92F04	valuation	verified
0x98315B281992097afc0c3A44b09e8Ea8bB5f3168	distribution	verified
0x377fA3430F1f6154EA5AFBc577fDdF67a5a9bA81	staking	verified
0x9ef078ee22ad58bcae3f6a2e423e0804a13ed699	referral	verified

Appendix B: Methodology and signal mapping

Methodology

This report is **Tier 1** automated tool output. The verdict is emitted upstream by the `token-scan-data-engine` (ASVS-style tiered checklist) and rendered verbatim by this framework — the framework does **not** compute a verdict locally. Per-module findings in §2 are reported from the same scan evidence.

Verdict construction (scanner-side)

Each scan produces two labels:

- **Subject verdict** — `HIGH` / `MIXED` / `LIMITED` / `LOW` / `INSUFFICIENT` — composed from per-axis outcomes by a boolean rule (no weighted score):
  - `any HARD_FAIL` → **LOW**
  - `≥ 2 SOFT_FAIL` → **LIMITED**
  - `1 SOFT_FAIL` → **MIXED**
  - all `PASS` (axes in `N_A` excluded from the denominator) → **HIGH**
  - `applicable / total < 0.5` → **INSUFFICIENT**

- **Analysis verdict** — `COMPLETE` / `PARTIAL` / `SPARSE` / `INSUFFICIENT` — reflects the share of evidence modules that ran successfully (independent of the subject verdict).

Axis outcomes

Each axis carries a **modality** and an **outcome**:

- Modality: `MUST` (dealbreaker — any HARD_FAIL drives Subject verdict to LOW) or `SHOULD` (best practice — accumulates SOFT_FAILs).
- Outcome: `PASS` / `SOFT_FAIL` / `HARD_FAIL` / `N_A`.

Profile-specific axis sets:

Profile	Axes
mainnet	`source_visibility`, `control_concentration`, `holder_concentration` (HHI; Hacken/antitrust convention <1500 deconcentrated, 1500–2500 moderate, ≥2500 concentrated), `liquidity_visibility`, `upgrade_authority`
testnet	`source_visibility`, `role_separation`, `upgrade_authority_separation`, `access_control_pattern` (deployment-readiness semantics)

Determinism and versioning

The verdict is a deterministic function of `(evidence, derived_signals, verdict_logic_version)`. The scanner bumps `verdict_logic_version` on every axis / threshold / composition rule change; the version applicable to this scan is recorded in Appendix C alongside the scanner version, so report vintage is fully traceable.

Reader caveat

The verdict reflects the share of the public on-chain transparency surface that the scan could assess against its codified checklist. It is **not** a merit ranking and **does not** assess whether the project is suitable for any particular use. A `LOW` label on a testnet scan with unverified source may reflect an operational choice rather than a quality problem with the contract.

Appendix C: Versions and timestamps

- Framework version: `1.2.0`
- Scanner version: `0.3.0`
- Verdict logic version: `6` (scanner-emitted)
- Scan id: `b6ad96b7-c1db-4ed4-8db1-c989f079fbbf`
- Scan started: `2026-05-14T15:54:52.554Z`
- Scan completed: `2026-05-14T15:57:46.233Z`
- Input source: `internal`

- Methodology references: `docs/methodology/04-smart-contract-methodology.md`, `docs/methodology/08-onchain-verification-methodology.md`, `docs/methodology/09-chain-specific-checks.md`, `docs/methodology/12-multi-contract.md`
- Schemas: `scan-evidence-v1`, `llm-pack-v1` (`docs/scan-service/schemas/`)

Appendix D: Disclaimer

This document is the output of an **automated tool** (xcactus Scan, Tier 1). It is generated without human review per request and is provided as a transparency aid for buy-side due diligence.

- **Not a security audit.** A full audit (Tier 2) is a separate, human-signed deliverable. Tier 1 does not replace it.
- **Not a MiCA-compliant deliverable.** MiCA compliance is assessed only within Tier 2 audits where the scope explicitly includes it.
- **Not investment advice.** This document does not recommend any action. Readers must form their own conclusions.
- **Not a guarantee.** Coverage is bounded by public on-chain evidence and the limitations enumerated in §3.

Findings reflect the state of the queried public data sources at the timestamps in Appendix C. Subsequent on-chain or off-chain changes are out of scope.