WebTrustEngine

R50 · v3.7 · Public

Technical Trust Note

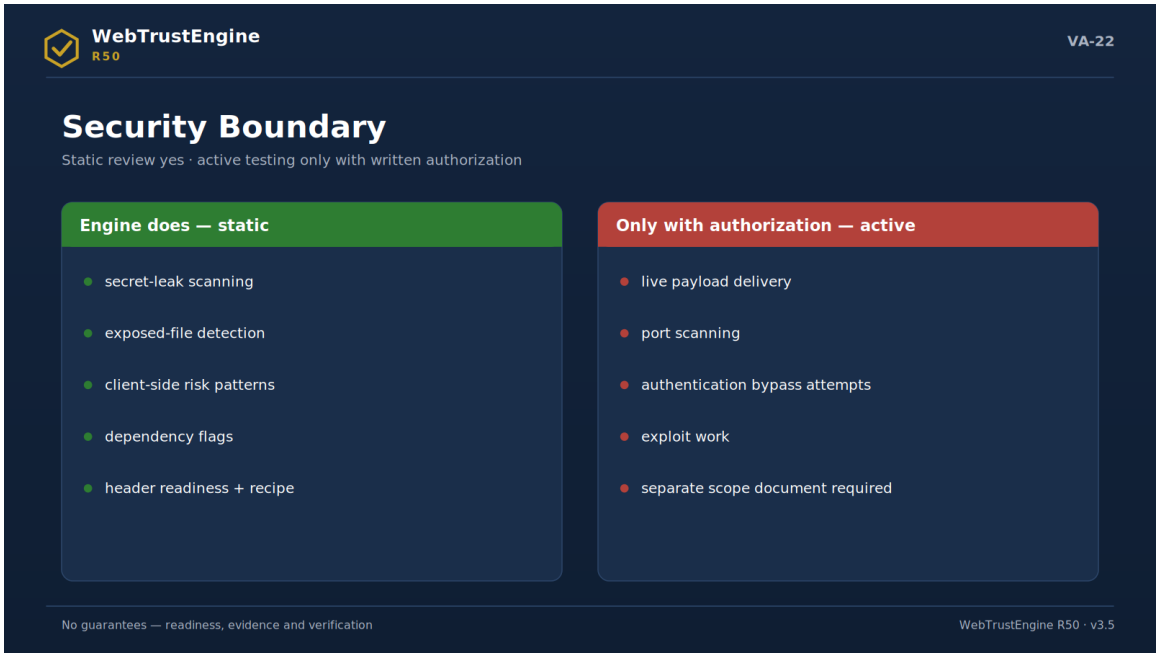

Figure VA-22 — The security boundary: static review is engine work; active testing needs separate authorization and scope.

Architecture, boundaries and the verification contract — for technical evaluators

July 2026

1. Purpose and Audience

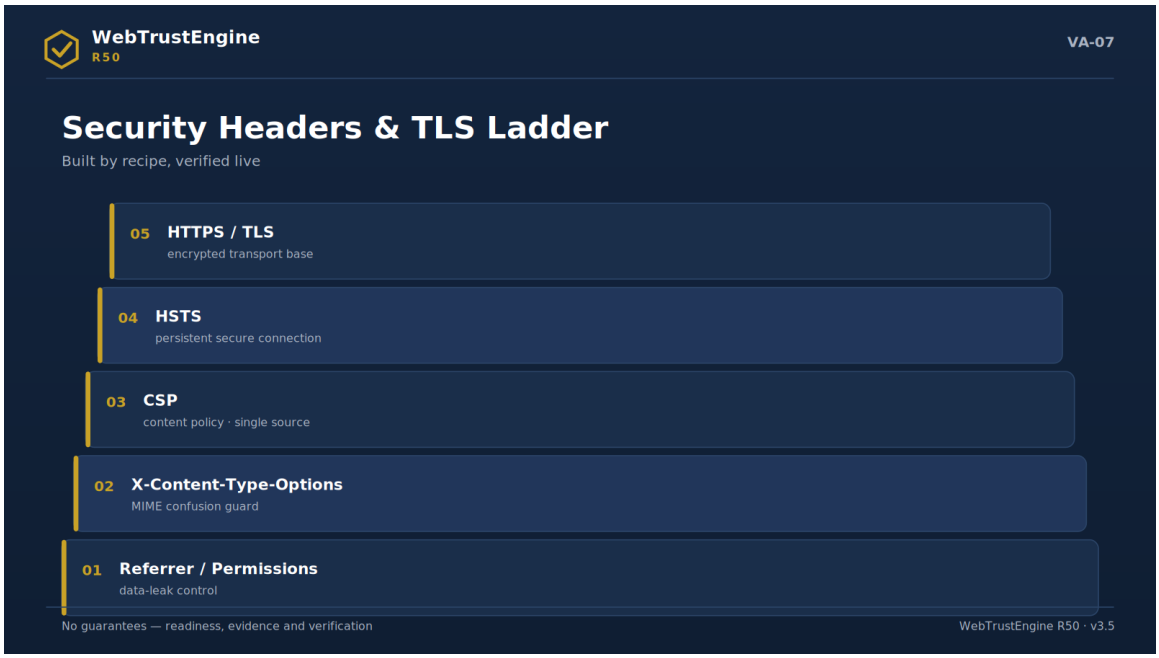

Figure VA-07 — The security headers and TLS ladder: from base transport to policy layer.

This note is written for CTOs, security teams, DevOps and technical procurement. The aim is verifiability, not marketing: what the engine does, what it deliberately does not, where the numbers come from and which file evidences each claim.

2. Architecture Overview

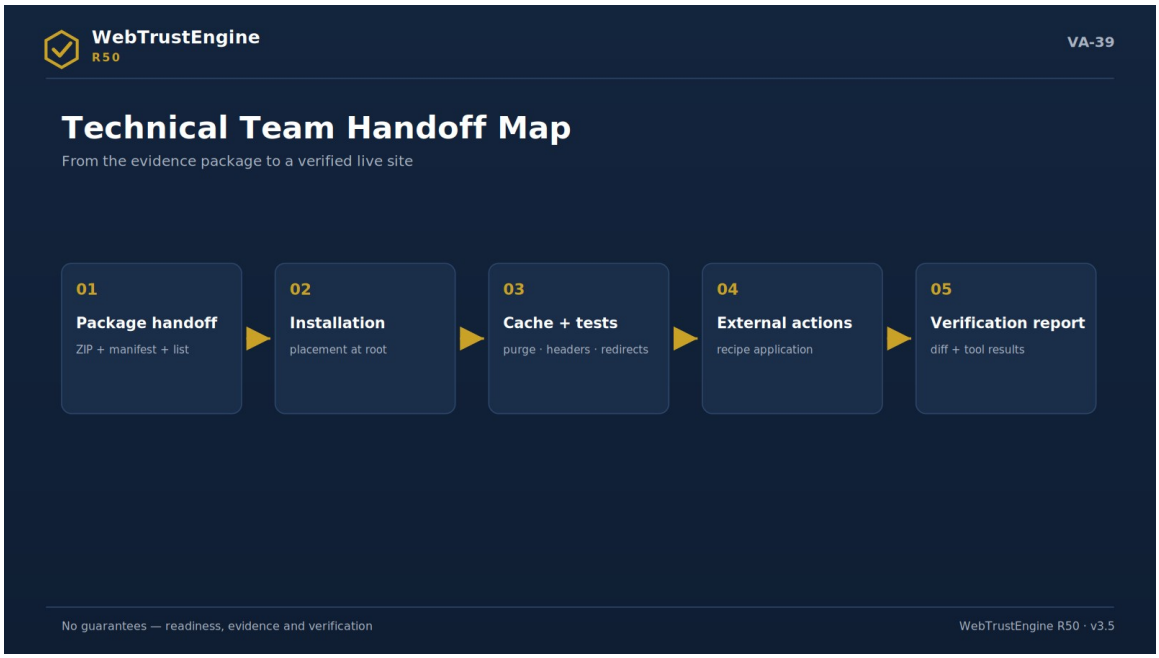

Figure VA-39 — The handoff map: five steps from delivery to the verification report.

The flow has five stages: input (ZIP/folder/authorized URL) → static analysis (319 detections + 68 security patterns) → SafeFix production (26 generators, on a working copy) → evidence package (score/manifest/rollback/checksums) → verification (21 runtime bridges + 24 external recipes + Deploy-Verify). The frozen core does not change between releases; new capability lands in a separate layer.

3. Input Types and Their Limits

ZIP

Shows file truth; cannot show server behaviour. The fastest, safest mode.

Local folder

Fast iteration on a dev copy; the pre-launch gate.

Live URL

Shows response truth; only with ownership/authorization and read-only access.

4. The Static Detection Layer: How 319 Is Counted

319 = 80 core + 239 granular. Counting is code-derived (check identifiers) and cross-verified against the CLI list, the quality gate and the registry; the three must agree. One check may functionally cover multiple catalog rows; hence catalog-row counts are never reported as check counts.

5. Security Layer Detail

Four blocks: (1) secret scanning — key/password/token patterns; (2) exposure surface — .env, backups, .git, config; (3) client risks — dangerous sinks, eval/document.write, mixed content; (4) server-side SAST classification — SQLi/command/traversal/deserialization/SSRF/XXE markers. SCA/CVE logic flags risky versions and binds them to update recipes; no 'exploitable' claims are made. All findings map to OWASP language. The engine produces no exploit code; any DAST-class behaviour requires written authorization plus scope.

6. SafeFix Engineering

Principles: working-copy isolation; idempotent generators (a second identical run yields no diff); file-level diff; one-step rollback (backup dir + changed/created lists); behaviour-neutral security (no CSP-breaking inline tricks; no duplicate policy when an existing one is detected); approval-class content untouched. Every run emits FIX_MANIFEST and ROLLBACK_MANIFEST.

7. The Runtime Bridge Contract

The contract: for every check not exactly measurable statically, the bridge record carries the quadruple {check, tool, metric, how}. In offline runs the status returns 'requires_live' — the engine fabricates no values. In the live phase the independent tool measures; the bridge binds the result to baseline and report. The 21 bridges cover header grades, TLS, lab/field performance, contrast, redirect chains and preview inspectors.

8. The External Recipe Template

Every recipe carries three fields: where (panel/record/service), steps (ordered and exact), verification (which query confirms it). The 24 headings cover DNS (DNSSEC, CAA), email identity (SPF/DKIM/DMARC/MTA-STS), CDN/WAF, HSTS preload, search consoles, server headers and security.txt. The engine never marks these 'done'; execution belongs to the owner, verification runs via bridges.

9. Counter Consistency and Quality Gates

Release acceptance hinges on three equalities: CLI list = quality gate = registry. Counting scripts exclude summary lines; off-by-one-class errors are caught by the gate and the fix is recorded in release notes. Test suites (engine + security) run each release; the result report enters the evidence package.

10. Evidence and Integrity Discipline

After packaging, SHA-256 digests are produced from real files; the manifest carries the file/size/type inventory. Delivery verification is three steps: unzip, verify checksums, confirm presence of critical files (scores, rollback, recipes). This buyer-side procedure takes five minutes and reduces vendor claims to file truth.

11. Known Limits

- Static analysis cannot see server-runtime behaviour; live verification is the mandatory complement
- JS-rendered content is partially visible statically; live bridges are advised on critical pages
- Field CWV is not produced; it is CrUX/PageSpeed-sourced
- Active security testing is out of scope; an authorized engagement is required
- Legal-compliance judgment sits in the human layer

12. The Verification Matrix

Claim	Evidence file	Independent confirmation
319 working checks	CONTROL_REGISTRY + CLI list	registry row count
Changes reversible	ROLLBACK_MANIFEST	a reversal drill

Headers ready	fixed ZIP .htaccess	SecurityHeaders (live)
TLS readiness	redirect+HSTS recipe	SSL Labs (live)
Perf readiness	defer/cache changes	PageSpeed (live)
Integrity	CHECKSUMS.sha256	sha256sum -c

This matrix is designed so a technical evaluator can perform all verification independently; no row rests on vendor assertion.

13. Input Preparation Guide

The ZIP must be taken from the root (index file at root) and include all asset folders (css/js/img/fonts). Exclude: node_modules, .git history, local backups and private key files — these create analysis noise and unnecessary secret exposure. Symlinks should be dereferenced; empty folders are fine. On multilingual sites, language copies must sit in the same ZIP at their real paths so the hreflang audit can run.

14. The Finding Schema

Every finding carries five fields: module (detector id), severity (info/medium/high), file (relative path), axis (SEO/A11Y/TRUST/PERF/SCHEMA/CONTENT/DELIVER), class (ENTEGRE/RUNTIME/EXTERNAL/CONDITIONAL/MANUAL). This schema enables filtering, prioritisation and class-based routing from one list. Severity expresses signal weight, not fix urgency; urgency is read together with the domain score and business context.

15. The Scoring Model

A domain score derives out of 100 by normalising finding density and severity mix in that axis against page count. Overall is the mean of domains. Deliberate design choices: no single pass/fail stamp (the decision belongs to context); the score is an internal-readiness measure never conflated with independent tool grades; cross-run comparison is meaningful only with the same input type and scope.

16. Anatomy of the .htaccess Recipe

Recipe blocks in order: (1) HTTPS redirect + HSTS; (2) security headers (nosniff, frame-ancestors/X-Frame, Referrer-Policy, Permissions-Policy); (3) CSP — only when the site lacks its own policy; (4) compression; (5) cache-control lifetimes by asset type; (6) directory listing off. Each block is comment-tagged; if hosting refuses one, it can be disabled per block.

17. CSP Conflict Prevention

The flow: existing-policy detection (server config files + meta tags scanned) → if present, no engine-side CSP is added, only a compatibility note → if absent, an origin-aware baseline is proposed (used external

origins whitelisted) → inline-script-breaking tricks are never produced. Because double policies intersect and narrow, the single-source principle is absolute.

18. Multilingual and hreflang Rules

Audited: reciprocal hreflang pairs per language copy, x-default presence, canonicals pointing within their own language, language attributes and per-language completeness of meta sets, cross-language consistency of identity signals. One-way hreflang, cross-canonicals and mixed-language meta are the three most frequent errors, flagged at file level.

19. JSON-LD Audit Rules

Rules: every block is parsed (invalid JSON is its own finding); @type is matched to page context; per-type required fields are audited (Organization→name/url, FAQPage→Q-A pairs, Breadcrumb→ordered items); cross-block identity consistency (same name/URL/sameAs) is verified; sector types are suggested as conditional rules only when content exists. Production is cautious under the principle that wrong schema costs more than missing schema.

20. The Static Performance Rule Set

Applied: defer on external scripts (inline untouched); flagging images missing width/height; lazy suggestions for offscreen images; compression + type-based cache lifetimes for text assets; @import chains and giant single-CSS warnings; a third-party script inventory. Not applied: automatic minify rewrites (source integrity) and critical-CSS extraction (render-dependent) — reported as recommendations.

21. The Deploy-Verify Technical Procedure

Seven steps: (1) extract the fixed ZIP at root; (2) hosting cache purge + CDN purge (if any); (3) fetch live headers and compare line-by-line with the expected set; (4) trace HTTP→HTTPS and the www decision on three sample URLs; (5) verify 404 behaviour with a non-existent URL; (6) refresh two representative pages in OG/Twitter debuggers; (7) run the independent tool pass and bind bridge outputs to baseline. The report yields the expected ↔ live diff table.

22. Data Handling and Confidentiality

Input (ZIP/folder) is processed solely for evaluation; production outputs are limited to the delivered package. Secret-scan findings enter reports masked (full values never written). Live-URL mode is read-only; authenticated areas are not entered. Retention and disposal are defined in the scope agreement.

23. Integration Points

The engine runs from the command line, which makes three integrations natural: (1) a CI gate — running Review on the release branch against a score threshold; (2) scheduled Monitor — periodic diffs via cron; (3) the delivery pipeline — automatic Review of agency output before acceptance. Outputs are JSON+Markdown, so they feed existing reporting tools.

24. The Run Reference

A typical run takes the input path, the output directory and a mode flag; an apply flag is added when fixes are to be written, and a page cap splits runs on large sites. The output directory contract is fixed:

- site_work/ — the working copy with fixes applied
- reports/ — score + finding + class reports
- safefix/ — FIX_MANIFEST + ROLLBACK_MANIFEST + backups
- RUN_SUMMARY.json — the run's at-a-glance record

Re-running the same input yields no new diff by idempotence; if it does, that is an error signal and halts release acceptance.

25. Output Schemas — Field by Field

RUN_SUMMARY.json

Field	Meaning
input / mode	input path and run mode
pages	pages processed
score_before / score_after	internal readiness scores
fixes_applied	count of applied fixes
frozen_engine	core version signature
outputs	produced file paths

FIX_MANIFEST.json

Field	Meaning
schema_version / mode	schema version and run mode
change_count	total changes
needs_human_count	items awaiting human approval
backup_dir	path of backups
changes[]	records at type+file+note level

ROLLBACK_MANIFEST.json

Field	Meaning
backup_dir	the source of reversal
files[]	changed files
created_files[]	newly created (removed on reversal)

26. Failure Modes and Diagnosis

Mode: Partial ZIP

Symptom: Findings abnormally dense; asset-missing warnings cluster.

Fix: Re-take the full ZIP from root; review the exclude list.

Mode: Encoding issue

Symptom: Turkish characters garble; content findings skew.

Fix: Normalise sources to UTF-8; re-run.

Mode: Permission/read-only output

Symptom: SafeFix cannot write; the manifest is empty.

Fix: Move the output directory to a writable location.

Mode: Policy clash

Symptom: Live style/script breakage; CSP violations in console.

Fix: Single-source rule: with a server policy present, engine CSP is removed; inline hacks never used.

Mode: Cache illusion

Symptom: Files correct, live stale.

Fix: Run the purge step; repeat the header fetch.

Mode: Preview staleness

Symptom: Social cards show days-old state.

Fix: Debugger refresh; verify with two representative URLs.

27. The Release Acceptance Checklist

- CLI = gate = registry (80/26) equality
- Test suites (engine+security) OK
- Idempotence: second run yields zero diff
- Full pass on two reference sites
- ROADMAP = 0 confirmed
- Patch/Count/Test reports produced
- Checksums computed post-packaging
- Open-verify: checksums + critical-file presence
- Freeze snapshot unchanged
- Claim-safety grep clean
- A rollback drill performed
- The number ledger current

28. The Technical Mini-Glossary

- Idempotency — A re-run on the same input producing no new diff.
- Freeze snapshot — The core signature unchanged across releases.

- Axis — The finding's quality axis (SEO/A11Y/TRUST…).
- Class — The finding's artifact class (ENTEGRE/RUNTIME/…).
- requires_live — The bridge's offline status value.
- Maskeli bulgu — Secret values never written to reports.
- Tek-kaynak CSP — The rule of managing policy from one source.
- Beklenen set — The header list live fetches compare against.
- Unzip-verify — Post-package unzip+checksum+critical-file check.
- Diff raporu — The expected ↔ live difference table.

29. Anatomy of the Two Reference Runs

Each release is accepted via full passes on two reference sites; the R50 runs profiled as follows (scores are the engine's internal measure, not live tool grades):

Run	Internal score (before→after)	Leading fix classes
Reference A (corporate product site)	88 → 96	header recipe · OG/Twitter completion · sitemap/robots refresh
Reference B (corporate services site)	87 → 98	meta/canonical completion · baseline JSON-LD · image dimensions/lazy

The shared pattern of the two runs is instructive: the largest score jumps come from the 'missing meta layer' and 'delivery hygiene' classes; security headers change the risk profile rather than the score, and their real counterpart appears only in the live pass. Run outputs (RUN_SUMMARY, both manifests, changed-file lists) are retained in the release evidence package.

30. A Bridge-Result Binding Example

One line shows the bridge in practice. The expected-set record says: 'the strict-transport-security header must be present in the live response; measurement: live header fetch + SecurityHeaders'. In the Deploy-Verify pass the live fetch returns one of three outcomes: the header present with the expected value (the row closes), present with a different value (written to the diff table as a value mismatch), or absent (purge check first, then server-processing diagnosis). The bridge's job is binding this three-way outcome to the baseline row — so the 'header prepared' claim sits together with the date and response that verified it.

31. Recipe-Verification Query Patterns

The 'verification' field of external recipes rests on standard admin queries; the most used patterns:

- Email identity: a TXT lookup showing the SPF/DMARC text is live
- DNSSEC: the DS lookup not returning empty
- HSTS preload: the domain visible on the preload status page
- Server headers: fetching live response headers and comparing with the expected set
- security.txt: the /.well-known/security.txt path returning 200
- Search consoles: property verification showing as 'verified' in the panel

What the patterns share is neutrality: verification rests not on the recipe author's assertion but on a query anyone can repeat.

32. The Guide Cross-Reference Map

Each main topic of this note maps to a wider section in the 46-page Capability Guide; a map for readers who want depth:

Topic in this note	Guide section
Architecture and modes (§2-3)	Guide §5-9 (ten domains + four modes deep)
The security layer (§5)	Guide §11 Security Boundary
SafeFix engineering (§6)	Guide §7 + §31 Reading the Reports
The bridge contract (§7, §30)	Guide §18 Runtime Bridges
External recipes (§8, §31)	Guide §17 + §34 Tool Protocol
Failure modes (§26)	Guide §29 Common Mistakes
The acceptance checklist (§27)	Guide §32 Release Discipline
Data handling (§22)	Guide §16 Privacy + §38 Vendor Questions

The map makes visible that the two documents are parts of one body: the note carries the contract, the guide carries the context.

33. The Expected-Set Reference: Nine Headers

The nine headers Deploy-Verify compares line by line; the expected pattern and a one-line rationale for each:

Header	Expected pattern	Rationale
strict-transport-security	max-age + includeSubDomains	HTTPS permanence
content-security-policy	single source; origin whitelist	narrows the injection surface
x-content-type-options	nosniff	stops MIME confusion
x-frame-options / frame-ancestors	SAMEORIGIN or the CSP equivalent	clickjacking protection
referrer-policy	strict-origin-when-cross-origin	limits URL leakage
permissions-policy	unused APIs off	surface reduction
cache-control (assets)	type-based max-age	speed vs freshness balance
content-type + charset	correct type; utf-8	encoding consistency
location (redirects)	one hop; https target	no chains, no loops

The patterns are a starting reference; site-specific policy (e.g. third-party requirements) may deliberately alter the expected set — the reason is noted in the diff report.

34. Evidence-Package Acceptance Drill — A Five-Minute Buyer Procedure

A technical evaluator can verify the delivered evidence package by hand in five minutes; this drill needs no vendor assertion and rests each step on a repeatable query.

- Step 1 — Open and inventory: unzip and compare the MANIFEST file/size/type inventory against the expected structure. A missing top folder (reports/, safefix/) is the first warning.

- Step 2 — Verify integrity: run the standard verification command over the checksum file; all lines must return OK. Even one FAIL means the package was altered and is rejected.
- Step 3 — Audit the number: compare the CONTROL_REGISTRY (or equivalent) row count with the marketing number (319); the catalog size (2,033) must sit in a separate column, the two never conflated.
- Step 4 — Test reversal: copy one file back from the ROLLBACK_MANIFEST backup_dir and confirm the site opens — this is the live drill of the 'reversible' claim.
- Step 5 — Check the live binding: in the Deploy-Verify diff report, verify at least one header row's expected ↔ live comparison and its bound independent-tool result; a 'prepared' claim must attach to a date and a response.

If the drill passes, the package is accepted; if any step fails, the problem class (integrity / number / reversal / live binding) is clear and can be conveyed to the vendor in one sentence. The procedure itself is the buyer-side proof of the product's 'the files speak' promise.