WebTrustEngine

R50 · PUBLIC LAUNCH EDITION · v3.7

Public Capability Guide

Evidence-first web governance, safe improvement and post-deployment verification

WebTrustEngine treats a corporate web asset not as a one-off design task but as a measurable, verifiable and improvable digital asset.

July 2026 · Public

Table of Contents

▶ 39. First-Review Prep

1. Executive Overview

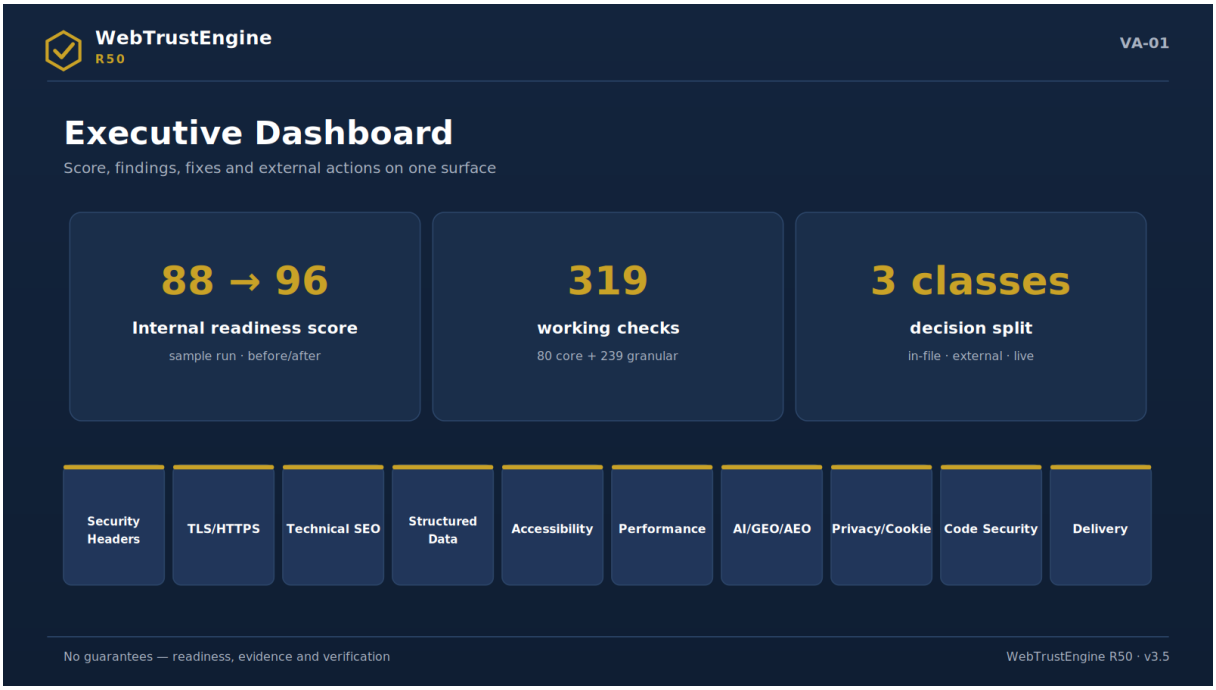

Figure VA-01 — Executive dashboard: internal readiness score, classified findings and the decision split on one surface.

A corporate website is no longer a one-off design project. The moment a web asset goes live, it is continuously read by search engines, AI answer systems, security scanners, accessibility tools, social-preview platforms and browser security policies. None of these readers care about aesthetics; they read headers, metadata, schemas, certificates, policies and response behaviour.

That is why web governance is needed. Governance is not fixing a site once; it is a repeatable discipline of measuring, safely improving, evidencing change and verifying it live. Scattered tool scores do not form a decision surface on their own; the executive wants an answer to 'which risk closed, which file changed, what remained external'.

WebTrustEngine is not a 'website fixer'; it is an evidence-first web governance engine. Review produces the baseline snapshot, SafeFix produces a low-risk reversible improvement package, Build creates new static surfaces, and Monitor / Deploy-Verify ties the live effect to evidence. Every step outputs a score, a changed-file list, a rollback manifest, external-action recipes and a verification plan.

R50's distinguishing trait is number discipline. The engine works over a 2.033-item reference catalog, yet never presents that number as 'automated checks'. The actual working detection layer is kept separately as 319 implemented detectable checks. Every catalog item is attached to the correct artifact class: working check, runtime bridge, external-action recipe, conditional rule, manual verification or a not-applicable rationale. Unclassified items — ROADMAP — are zero.

This guide is a single public source for the executive to decide, for marketing to speak with the right language, and for the technical team to see the boundaries clearly. It contains no pricing, offers or customer-specific information, and claims no live independent tool results.

What this provides: Not just a score: a decidable risk map, an evidence surface and an honest boundary set.

2. The Web Governance Problem

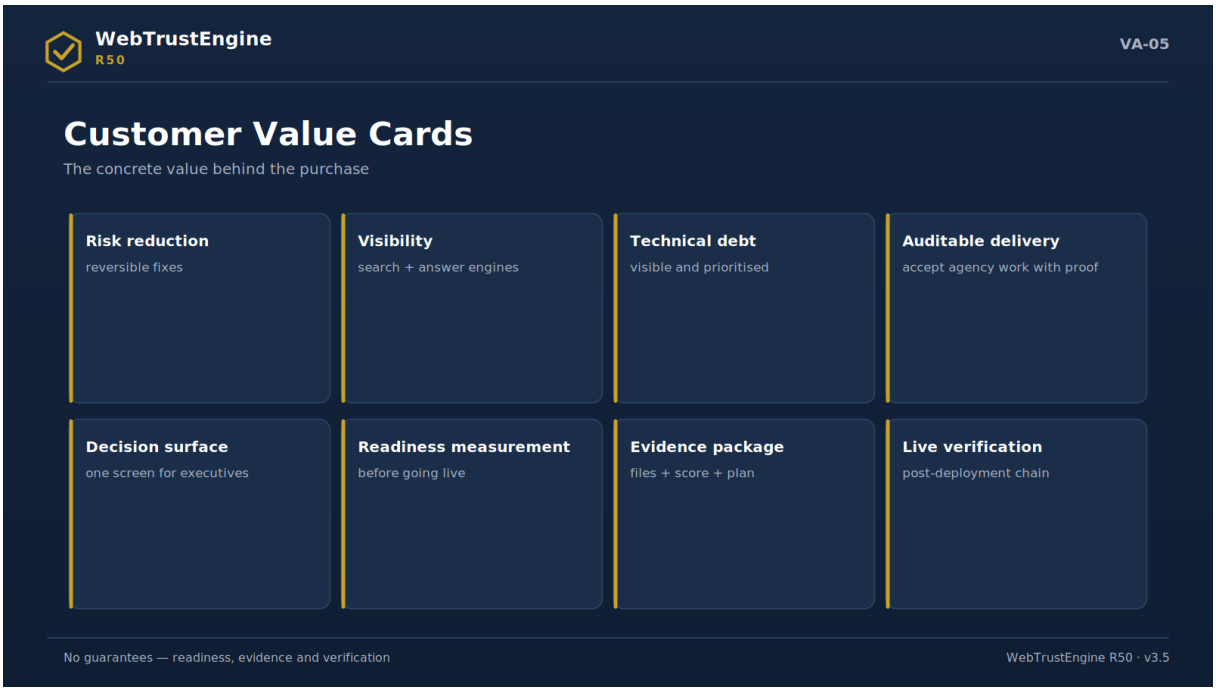

Figure VA-05 — Customer value cards: eight concrete values from risk reduction to live verification.

A website is now a human storefront plus a machine surface.

The page a visitor sees is simultaneously the input of dozens of automated readers. A page flawless for humans can carry missing headers, broken schemas and unclear identity signals for machines.

Even with great design, machine readability can be weak.

Visual refresh projects often never touch the meta layer, security headers or structured data; the site looks 'new' but stays old on the audit surface.

Ownership is scattered across teams.

SEO with an agency, security with IT, accessibility with no one, social preview with marketing. Scattered ownership produces scattered scores.

Scattered tool scores do not give the executive a decision surface.

Five scores from five tools do not answer 'what should we do and what changed'. Decisions need one framework and one evidence chain.

WebTrustEngine reduces this scatter to one report and one change proof.

The 10-domain score model, changed-file list, rollback manifest, external-action recipes and the Deploy-Verify loop unite in a single frame.

Scattered State	WebTrustEngine Approach
Disconnected scores from different tools	10-domain score model
Unknown what changed	Changed files + rollback manifest
Unclear if it works live	Deploy-Verify
Hosting/DNS work gets mixed in	External action recipe
Automation appears to do everything	Manual / runtime / external separation

3. R50 Capability Snapshot

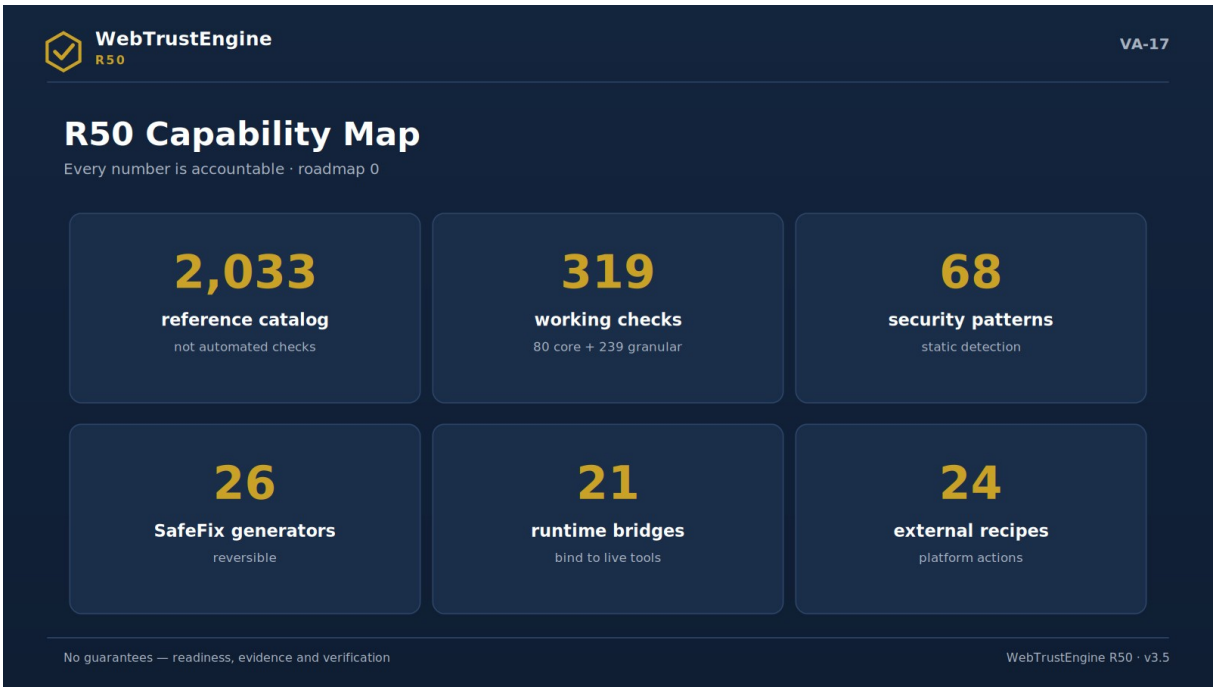

Figure VA-17 — The R50 capability map: six core numbers and zero open roadmap items.

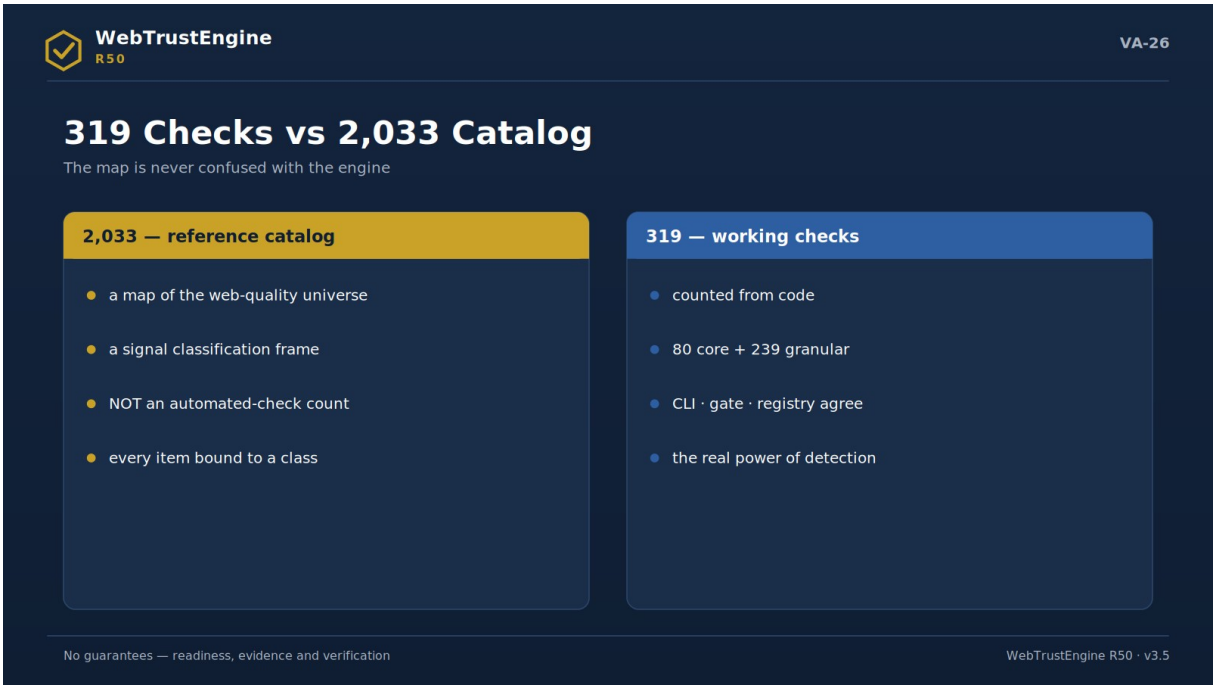

Figure VA-26 — The 319 vs 2,033 split: the map (catalog) is never confused with the engine (working checks).

Each number below is explained with three questions: what it means, what it does not mean, and why it must be told carefully in a public document.

2.033 Reference Catalog Items

What it means: The classification map of signals that external tools, standards and web-quality domains can report; the reference universe where the engine places findings and boundaries.

What it does not mean: It is not the number of automated checks; the sentence 'the engine runs 2,033 checks' is not an accurate reading and does not appear in any material.

Why it matters: If this number turns into an inflated claim in public text, the product's strongest trait — honesty discipline — is damaged; hence every mention states it is a reference catalog.

319 Implemented Detectable Checks

What it means: The total of detection checks the engine actually runs on files and code (80 core + 239 granular).

What it does not mean: It is not a catalog row count; one check may functionally cover multiple catalog rows.

Why it matters: This is the correct 'working checks' number for marketing copy; any other number is either short or inflated.

80 Core Checks

What it means: The core audit set: fundamental detections for meta, canonical, security-header readiness, schema, accessibility and delivery hygiene.

What it does not mean: It is not the whole engine; without the granular layer the detailed coverage is incomplete.

Why it matters: CLI, gate and registry agree on this core count (80); it is the public proof of counter consistency.

239 Granular Checks

What it means: Extended detail checks on top of the core: ARIA patterns, schema type fields, social-preview details, static performance markers.

What it does not mean: Not a separate product; together with the core it forms the single implemented layer.

Why it matters: It explains where detailed coverage comes from; it answers 'where does 319 come from'.

68 Security Patterns

What it means: Static security and risk patterns: secret scanning, exposed files, client-side risky patterns, server-side SAST classifications, malware markers, weak crypto.

What it does not mean: It is not active vulnerability testing (DAST); no live exploitation, port scanning or payloads.

Why it matters: To avoid a pentest perception, this layer is always described as a 'static security review'.

26 SafeFix Generators

What it means: Low-risk, reversible fix generators; they apply changes such as meta/OG/JSON-LD/sitemap/robots/header recipes on a working copy.

What it does not mean: Not a rebuild transformer; it does not enter risky structural changes.

Why it matters: This answers 'will the engine break my site': every change is reversible via the rollback manifest.

21 Runtime Bridges

What it means: Bridges to real tools for checks that cannot be exactly measured from a static file: PageSpeed, SSL Labs, SecurityHeaders, contrast, redirect chains.

What it does not mean: Not measurements the engine invents; offline it only states which tool will run.

Why it matters: The honest answer to 'does the engine produce live scores': no, it bridges.

24 External Action Recipes

What it means: Instruction sets for steps the engine cannot do in-file yet which affect quality: DNSSEC, SPF/DKIM/DMARC, Cloudflare, HSTS preload, Search Console.

What it does not mean: Not work the engine claims to have 'auto-fixed'; execution belongs to the owner.

Why it matters: Clearly stating the hosting/DNS/account boundary is part of the product's trust language.

53 Tools + 6 Standards

What it means: The reference base findings rest on: SecurityHeaders, Observatory, SSL Labs, PageSpeed, Lighthouse, CrUX, Search Console, axe-core; OWASP, WCAG 2.2, Schema.org, Core Web Vitals, W3C, llms.txt.

What it does not mean: The engine does not replace these tools; it classifies their signals and bridges for verification.

Why it matters: Emphasising independent verification prevents a 'self-scoring tool' perception.

ROADMAP = 0

What it means: It means no catalog item remains unclassified; every item is attached to an artifact.

What it does not mean: It does not mean 'everything is automated'; part sits in runtime, external action, conditional rules or manual verification.

Why it matters: Explaining ROADMAP=0 correctly makes the product's most easily misread claim safe.

In WebTrustEngine R50 the number 2,033 is not the count of automated checks. It is the reference catalog classifying signals produced by external tools, standards and web-quality domains. R50's credibility comes from never turning this number into an inflated claim. The engine's working detection layer is kept separately as 319 implemented detectable checks.

4. Catalog Status Model

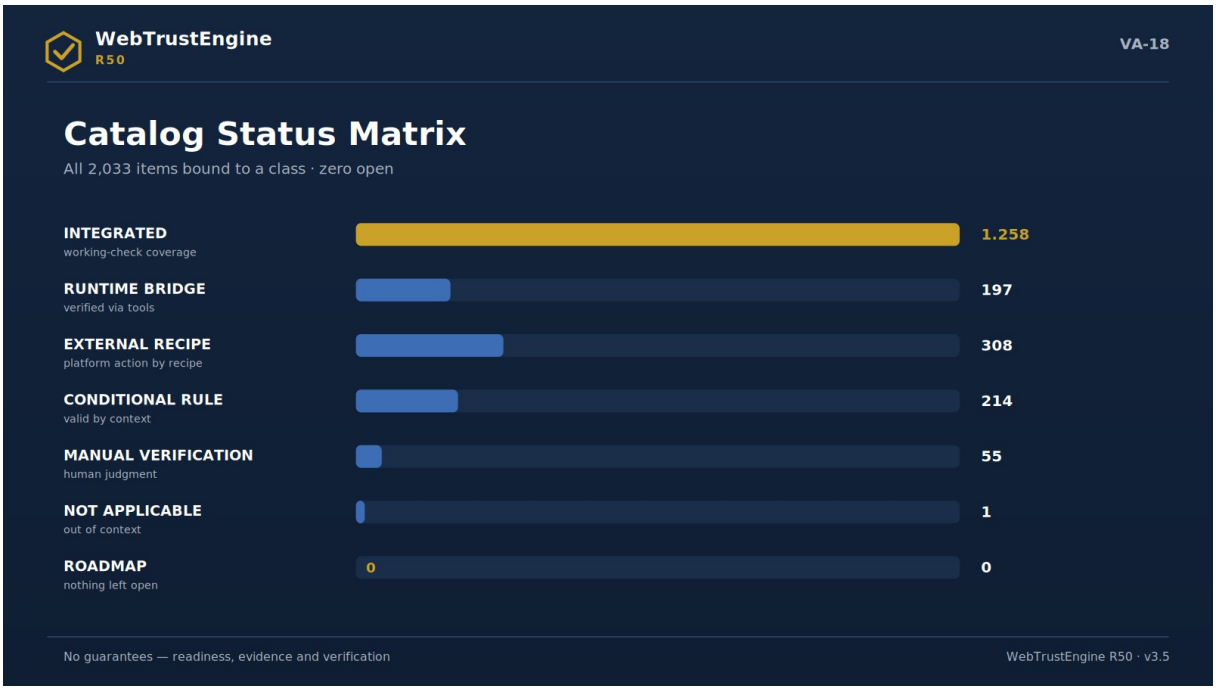

Figure VA-18 — The catalog status matrix: class distribution of 2,033 items and zero open entries.

The catalog status model is the visible form of R50's honesty architecture. The goal is not to throw every signal into two crude buckets of 'done' or 'to-do', but to attach it to the artifact class that fits its nature. A header policy can be produced in a file; a TLS grade can only be measured live; a DMARC record can only be enabled in a DNS panel; a contrast decision usually needs a human eye.

ROADMAP=0 carries a specific meaning in this model: it does not mean everything was automated; it means everything was attached to the right artifact class. This nuance is deliberately preserved in public communication, because a misread 'zero open items' easily becomes an inflated claim.

The table below gives the R50 distribution of the 2.033-item catalog, the public meaning of each status and typical examples.

Status	Count	Public meaning	Example
ENTEGRE	1,258	Signal functionally attached to working check/fix logic	title/meta/canonical/schema
RUNTIME-BRIDGE	197	Requires live tool or runtime verification	PageSpeed, SSL Labs, Observatory
EXTERNAL-RECIPE	308	Requires hosting/DNS/CDN/panel action	DMARC, DNSSEC, Cloudflare
CONDITIONAL-RULE	214	Valid when page type/sector applies	Product, Event, FAQ schema
MANUAL-VERIFICATION	55	Requires human/expert verification	some WCAG, legal claims, visual context
NOT-APPLICABLE	1	Not applicable in a static-site context	edge case
ROADMAP	0	No unclassified items remain	—

ROADMAP=0 It does not mean everything was automated; it means everything was attached to the right artifact class.

5. 10-Domain Score Model

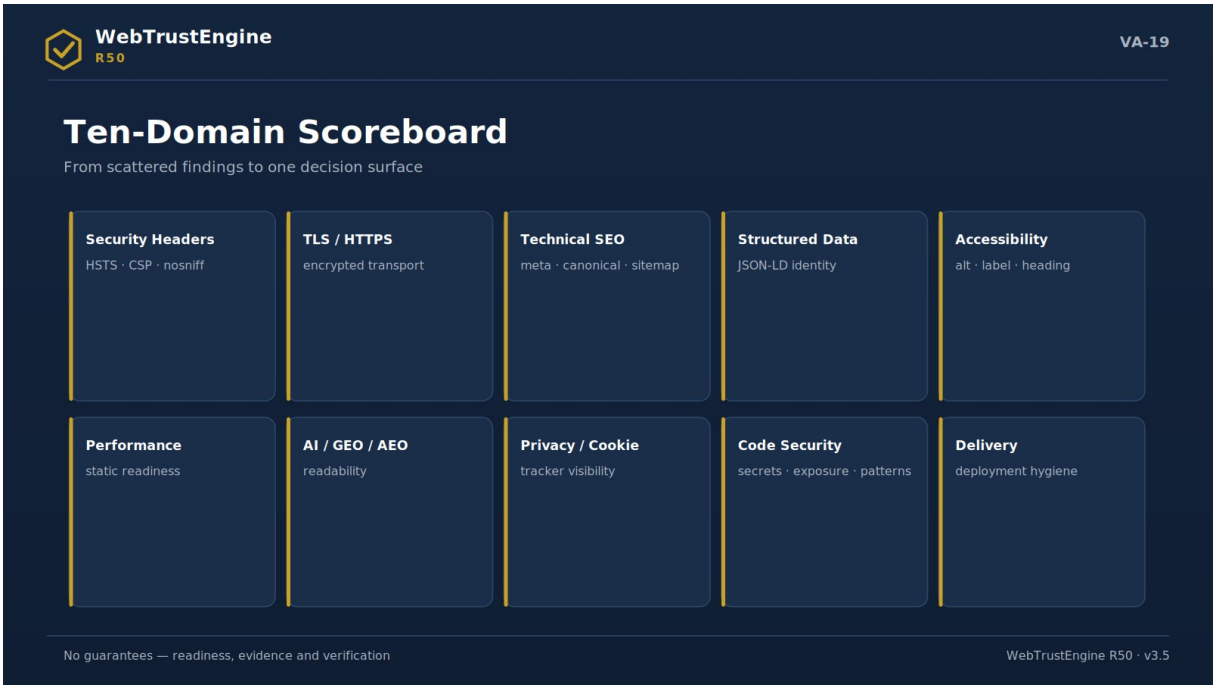

Figure VA-19 — The ten-domain scoreboard: the full decision surface from security to delivery.

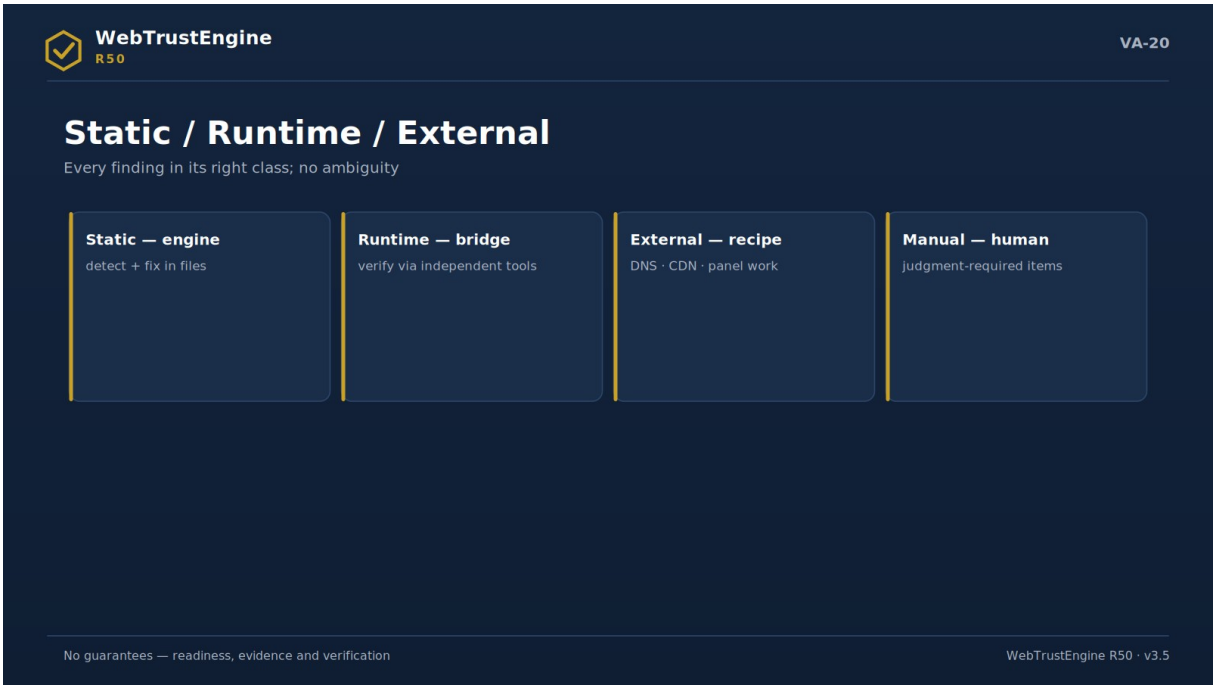

Figure VA-20 — The four-class split: static, runtime, external and manual verification.

Each domain follows the same seven-part template: purpose, typical problems, what the engine checks, what it can fix, live verification, sample output and executive meaning.

Security Headers

The Security Headers domain examines the policy layer that tells modern browsers within which security boundaries to run the page. It is not a mere 'is the header present?' check; HSTS, CSP, X-Frame-Options, X-Content-Type-Options, Referrer-Policy and Permissions-Policy must be handled in a way that fits the target site type.

Typical problems:

- Headers absent or on default server values
- CSP conflicts with inline code and breaks the page
- HSTS present but wrong max-age/scope
- Double CSP (server + file) intersects and narrows policy

What the engine checks: The engine statically classifies header presence, value quality, in-site conflicts (inline code ↔ strict CSP) and the risk of clashing with an existing server policy.

What it can fix: In supported environments (.htaccess) it produces a secure baseline header set; if the site already has its own CSP it does not add a duplicate policy; it turns advice into a recipe.

Live verification: Whether headers actually apply live depends on hosting behaviour; verified post-deployment with SecurityHeaders and Mozilla Observatory.

Sample indicator	Before	After
Header coverage	3/9	9/9
CSP conflict	var / present	yok / none
HSTS	yok / none	max-age + subdomains

What the engine does: The engine statically classifies header presence, value quality, in-site conflicts (inline code ↔ strict CSP) and the risk of clashing with an existing server policy.

What it doesn't: Whether headers actually apply live depends on hosting behaviour; verified post-deployment with SecurityHeaders and Mozilla Observatory.

What the output is: In supported environments (.htaccess) it produces a secure baseline header set; if the site already has its own CSP it does not add a duplicate policy; it turns advice into a recipe.

How it ports to the website: Ported to site copy as the twin sentence 'security-header readiness + independent post-deploy verification'.

Meaning for the executive: This domain answers 'what does our site promise the browser'; a low score is an invisible yet measurable corporate risk.

Meaning for the technical team: The gap between file production and live behaviour is most visible here; a post-deploy header fetch is mandatory.

TLS / HTTPS Readiness

This domain examines readiness for encrypted, secure transport: HTTPS enforcement, HSTS policy, mixed content and certificate readiness in one frame.

Typical problems:

- HTTP version still reachable
- http:// resources inside the page (mixed content)
- No HSTS or not preload-ready
- Form action goes over http

What the engine checks: The engine statically detects redirect setup, in-page http:// references, form targets and HSTS markers.

What it can fix: Produces an .htaccess recipe for HTTPS redirect + HSTS; flags mixed-content lines at file/line level.

Live verification: Certificate chain, protocol versions and OCSP stapling are graded live by SSL Labs; HSTS preload submission is an external action.

Sample indicator	Before	After
Mixed content	14 lines	0
HTTPS zorlama / enforcement	partial	tam / full
HSTS	yok / none	recipe ready

What the engine does: The engine statically detects redirect setup, in-page http:// references, form targets and HSTS markers.

What it doesn't: Certificate chain, protocol versions and OCSP stapling are graded live by SSL Labs; HSTS preload submission is an external action.

What the output is: Produces an .htaccess recipe for HTTPS redirect + HSTS; flags mixed-content lines at file/line level.

How it ports to the website: Ported as 'encrypted-transport readiness; live grade via independent tool post-deploy'.

Meaning for the executive: Browser 'not secure' warnings touch brand trust directly; this domain is the readiness that prevents them.

Meaning for the technical team: The TLS grade is decided server-side; the engine produces readiness, SSL Labs gives the grade.

Technical SEO

Technical SEO examines the technical base search engines need to crawl and index correctly: title/meta, canonical, hreflang, robots, sitemap, heading hierarchy and URL consistency.

Typical problems:

- Missing/duplicate titles and descriptions
- Canonical absent or self-contradictory
- Sitemap disagrees with the real page set
- Multiple or missing H1

What the engine checks: The engine statically audits per-page meta quality, canonical/hreflang consistency, robots-sitemap relation and heading structure.

What it can fix: Missing meta/canonical completion, sitemap/robots generation and heading fixes are applied via SafeFix on the working copy.

Live verification: Broken external links, redirect chains and real indexing need a full crawl; verified with Search Console.

Sample indicator	Before	After
coverage	%61	%100
Kanonik / canonical	12 eksik / missing	0
Sitemap	eski / stale	current

What the engine does: The engine statically audits per-page meta quality, canonical/hreflang consistency, robots-sitemap relation and heading structure.

What it doesn't: Broken external links, redirect chains and real indexing need a full crawl; verified with Search Console.

What the output is: Missing meta/canonical completion, sitemap/robots generation and heading fixes are applied via SafeFix on the working copy.

How it ports to the website: Ported as 'technical-SEO readiness — not a ranking guarantee'.

Meaning for the executive: This is the findability infrastructure; the precondition for content investment not being wasted.

Meaning for the technical team: There is no ranking promise; there is crawlability and consistency.

Structured Data

This domain examines the JSON-LD layer that lets machines understand content: correct type choice, required fields and entity consistency.

Typical problems:

- Schema absent or only on the homepage
- Required fields missing (name/provider/startDate)
- Schema mismatched to page type (e.g. Product everywhere)
- Identity signals (sameAs) disconnected

What the engine checks: The engine parses JSON-LD blocks and classifies missing required fields and type mismatch across Organization/WebSite/WebPage/Service/FAQPage/BreadcrumbList.

What it can fix: Produces baseline schema blocks and file-level missing-field warnings; sector types like Product/Event are separated as conditional rules.

Live verification: Rich-result eligibility is verified live with the Rich Results Test; wrong schema can break rich results.

Sample indicator	Before	After
pages with schema	%18	%100
required-field errors	23	0
sameAs	yok / none	tam / complete

What the engine does: The engine parses JSON-LD blocks and classifies missing required fields and type mismatch across Organization/WebSite/WebPage/Service/FAQPage/BreadcrumbList.

What it doesn't: Rich-result eligibility is verified live with the Rich Results Test; wrong schema can break rich results.

What the output is: Produces baseline schema blocks and file-level missing-field warnings; sector types like Product/Event are separated as conditional rules.

How it ports to the website: Ported as the trio 'entity clarity + correct type + rich-result verification'.

Meaning for the executive: Structured data is the brand's identity card in the machine world; when missing, the brand reads as 'anonymous'.

Meaning for the technical team: Schema correctness needs sync with content; conditional rules bind to page type.

Accessibility

The Accessibility domain examines readiness for users with disabilities and assistive technologies, deliberately separating static detection from human evaluation.

Typical problems:

- Alt texts empty or meaningless
- Form fields unlabeled
- Heading order skips (H1→H3)
- No landmarks/skip link

What the engine checks: The engine statically detects alt/label/ARIA usage, landmark and heading structure, language attribute and table semantics.

What it can fix: Flagging missing alt/label, landmark/heading fixes and the language attribute are within SafeFix scope.

Live verification: Contrast ratio, focus visibility, meaningful sequence and media description need render/human evaluation; routed to manual verification. No full-WCAG-compliance guarantee is given.

Sample indicator	Before	After
alt coverage	%64	%100
Etiketsiz form / unlabeled fields	9	0
Landmark	partial	tam / complete

What the engine does: The engine statically detects alt/label/ARIA usage, landmark and heading structure, language attribute and table semantics.

What it doesn't: Contrast ratio, focus visibility, meaningful sequence and media description need render/human evaluation; routed to manual verification. No full-WCAG-compliance guarantee is given.

What the output is: Flagging missing alt/label, landmark/heading fixes and the language attribute are within SafeFix scope.

How it ports to the website: The 'static readiness + manual verification' split ports verbatim to site copy.

Meaning for the executive: Accessibility is both inclusion and legal-risk territory; the readiness score makes both visible.

Meaning for the technical team: Items not auto-PASSed are an honesty layer, not a weakness.

Performance Readiness

This domain examines readiness to load fast via static indicators, deliberately keeping field measurement outside.

Typical problems:

- Render-blocking script/CSS
- Dimensionless images (CLS risk)
- No compression/cache header readiness
- Huge unoptimized images

What the engine checks: The engine statically classifies render-blocking patterns, image dimension/lazy signals, minify readiness and cache/compression markers.

What it can fix: Defer on external scripts, image dimension/lazy suggestions and an .htaccess compression/cache recipe are SafeFix outputs.

Live verification: Real Core Web Vitals (LCP/INP/CLS) are measured live via PageSpeed/CrUX; the engine produces no field data; CDN setup is an external action.

Sample indicator	Before	After
Render-blocking	7	0
dimensionless imgs	31	0
cache recipe	yok / none	ready

What the engine does: The engine statically classifies render-blocking patterns, image dimension/lazy signals, minify readiness and cache/compression markers.

What it doesn't: Real Core Web Vitals (LCP/INP/CLS) are measured live via PageSpeed/CrUX; the engine produces no field data; CDN setup is an external action.

What the output is: Defer on external scripts, image dimension/lazy suggestions and an .htaccess compression/cache recipe are SafeFix outputs.

How it ports to the website: Ported as 'static performance readiness — not CWV field measurement'.

Meaning for the executive: Speed is conversion's silent partner; the readiness score closes the first half of 'why are we slow live'.

Meaning for the technical team: The lab/field split is preserved: engine readiness, PageSpeed measurement.

AI / GEO / AEO

This domain prepares content for correct readability by AI systems, search and answer engines: identity clarity, sourceable text and machine directives.

Typical problems:

- Who/what/where unclear (weak entity)
- No llms.txt or AI directives
- FAQ not answer-ready
- Multilingual pages inconsistent

What the engine checks: The engine evaluates entity signals, llms.txt/AI-crawler directives, semantic structure, author/date markers and FAQ readiness.

What it can fix: Produces an llms.txt suggestion, structure for identity/service clarity and answer-ready FAQ markup.

Live verification: Being recommended/cited by AI is not guaranteed; readability readiness is produced and this boundary is written explicitly everywhere.

Sample indicator	Before	After
llms.txt	yok / none	ready
clarity	weak	strong
answer-ready FAQ	yok / none	marked

What the engine does: The engine evaluates entity signals, llms.txt/AI-crawler directives, semantic structure, author/date markers and FAQ readiness.

What it doesn't: Being recommended/cited by AI is not guaranteed; readability readiness is produced and this boundary is written explicitly everywhere.

What the output is: Produces an llms.txt suggestion, structure for identity/service clarity and answer-ready FAQ markup.

How it ports to the website: Ported as 'AI-readability readiness — no citation guarantee'.

Meaning for the executive: In the answer-engine era visibility is not just SERP but 'appearing inside the answer'; this domain builds that readiness.

Meaning for the technical team: Citation promises are banned; readiness signals are measurable and auditable.

Privacy / Cookie

This domain examines privacy readiness via tracker visibility and cookie hygiene, deliberately leaving legal-compliance judgment to the human layer.

Typical problems:

- Tracking markers before consent
- Cookie flags missing (Secure/HttpOnly/SameSite)
- Privacy/cookie policy unlinked
- Third-party trackers invisible

What the engine checks: The engine detects tracker visibility, cookie flags, pre-consent tracking markers and policy links (static signals parallel to the Blacklight/Cookiebot approach).

What it can fix: Produces cookie-flag warnings and policy-link suggestions.

Live verification: KVKK/GDPR compliance requires legal counsel; the engine gives no guarantee and routes to manual verification.

Sample indicator	Before	After
pre-consent	var / present	flagged

cookie flags	eksik / missing	recipe
policy link	yok / none	var / present

What the engine does: The engine detects tracker visibility, cookie flags, pre-consent tracking markers and policy links (static signals parallel to the Blacklight/Cookiebot approach).

What it doesn't: KVKK/GDPR compliance requires legal counsel; the engine gives no guarantee and routes to manual verification.

What the output is: Produces cookie-flag warnings and policy-link suggestions.

How it ports to the website: Ported as 'tracker visibility and cookie readiness — not legal advice'.

Meaning for the executive: Privacy is now reputation; it cannot be governed without visibility.

Meaning for the technical team: 'Compliance guarantee' language is banned; 'visibility + readiness + legal sign-off' is the correct chain.

Code / Static Security

This domain runs the static security review over source and configuration, clarifying the active-testing boundary from the start.

Typical problems:

- Hardcoded secrets (keys/passwords) in the repo
- Exposed files (.env, backups, .git)
- Risky JS patterns (eval, document.write)
- Weak-crypto markers (MD5/SHA1)

What the engine checks: The engine statically detects secret scanning, exposed-file checks, client-side risk patterns, server-side SAST classifications (SQLi/command/traversal/deser/SSRF/XXE), malware markers and weak crypto; findings map to OWASP.

What it can fix: Produces risky-pattern warnings, exposed-file markers and a security-header recipe; never produces exploit code.

Live verification: Active exploitation, port scanning, auth bypass and live payloads belong to a separate authorized security scope; this domain is not a pentest.

Sample indicator	Before	After
secrets	2	0
exposed files	5	0
mapping	yok / none	raporlu / reported

What the engine does: The engine statically detects secret scanning, exposed-file checks, client-side risk patterns, server-side SAST classifications (SQLi/command/traversal/deser/SSRF/XXE), malware markers and weak crypto; findings map to OWASP.

What it doesn't: Active exploitation, port scanning, auth bypass and live payloads belong to a separate authorized security scope; this domain is not a pentest.

What the output is: Produces risky-pattern warnings, exposed-file markers and a security-header recipe; never produces exploit code.

How it ports to the website: The pair 'static security review — not a pentest' enters site copy verbatim.

Meaning for the executive: This domain surfaces 'are unknown risks in the repo'; it is the dashboard of silent risk.

Meaning for the technical team: The SAST ↔ DAST split is written; no active behaviour without authorization.

Delivery / Deployment

This domain examines the hygiene of publishing correctly and consistently: favicon/manifest, 404, social preview, sitemap/robots consistency and file structure.

Typical problems:

- Favicon/manifest missing or inconsistent
- OG/Twitter cards half-done
- No 404 page
- robots contradicts sitemap

What the engine checks: The engine audits the icon/manifest set, social-preview fields, 404 presence and the robots-sitemap relation.

What it can fix: Completing favicon/manifest, generating OG/Twitter fields and robots/sitemap fixes are within SafeFix.

Live verification: Live redirect, header and preview effect are verified post-deploy; social-preview caches refresh via debuggers.

Sample indicator	Before	After
icon set	3 pcs	10 pcs
OG coverage	%40	%100
404	yok / none	custom page

What the engine does: The engine audits the icon/manifest set, social-preview fields, 404 presence and the robots-sitemap relation.

What it doesn't: Live redirect, header and preview effect are verified post-deploy; social-preview caches refresh via debuggers.

What the output is: Completing favicon/manifest, generating OG/Twitter fields and robots/sitemap fixes are within SafeFix.

How it ports to the website: Ported with the 'publish hygiene + Deploy-Verify' link.

Meaning for the executive: Delivery hygiene is the infrastructure of how the brand appears wherever it is shared.

Meaning for the technical team: Preview caches mislead; the live-verification step is never skipped.

6. Review Mode — Deep Dive

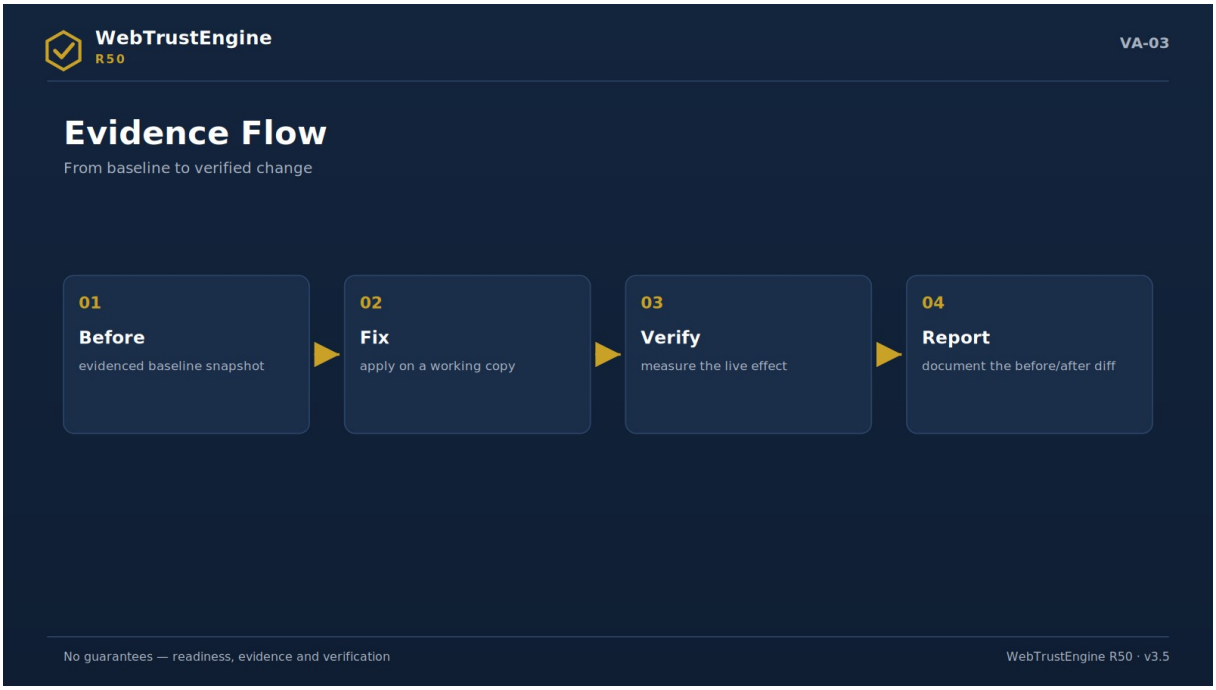

Figure VA-03 — The evidence flow: before, fix, verify, report.

Review is WebTrustEngine's decision surface that changes no files. It supports three input types: a ZIP package, a local folder and — with ownership/authorization — a live URL. ZIP review is ideal for seeing the integrity and hygiene of a delivered package; a local folder enables fast iteration on a development copy; live-URL review reads real response headers and publish behaviour, and is done only with authorization.

Which one to choose depends on the question itself: 'what did the agency deliver' wants a ZIP; 'where are we before go-live' wants a local folder; 'what actually returns live' wants a live URL. In all three modes Review modifies nothing; it only reads, classifies and reports.

The decision surface is built to answer: Where is the risk? Which domains score high? Which findings close via file fixes? Which need an external platform? Which need live verification? Which decision needs human approval? These six questions are what the executive wants on one screen.

Review outputs include the 10-domain score table, the findings list (domain/severity/file), the catalog-class distribution and the suggested next step (SafeFix scope + external-action list). Risk level derives from finding density and severity; there is deliberately no single 'pass/fail' stamp, because the decision belongs to context.

- ZIP: delivery audit
- Local folder: pre-launch iteration
- Live URL: only with authorization
- No file changes; a decision surface forms

What the engine does: Reads, classifies and scores by input type.

What it doesn't: Changes no files; never touches live without authorization.

What the output is: Score table + findings + class distribution + next step.

How it ports to the website: 'Run a Review' CTA plus the three input types go to Home/Review pages.

Meaning for the executive: A risk-decision map on one screen.

Meaning for the technical team: Produces the baseline; the reference for all later comparisons.

7. SafeFix Mode — Deep Dive

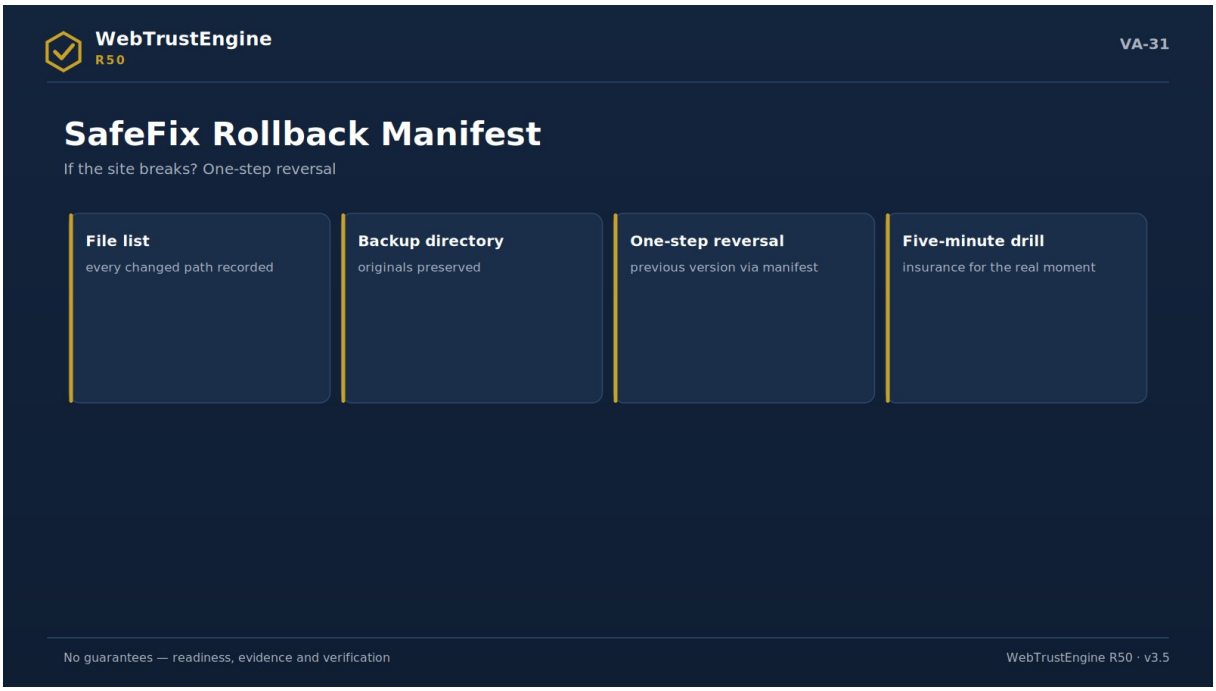

Figure VA-31 — The rollback manifest: file list, backup, one-step reversal and the drill.

SafeFix's philosophy is safe improvement, not rebuilding. The existing site is an asset: content, structure, link history. SafeFix preserves that asset and applies only low-risk, reversible changes on a working copy. 'Low risk' is measurable here: a change touches either the meta layer (title/description/OG/Twitter/JSON-LD), delivery hygiene (sitemap/robots/llms.txt/favicon), or behaviour-neutral security readiness (an .htaccess header recipe).

Every SafeFix run produces three proofs: the changed-file list, the rollback manifest and the before/after internal readiness score. The rollback manifest contains the backup directory, changed and created files — full one-step reversal is possible. This is the engineering answer to 'what if the engine breaks the site'.

Scope examples: missing meta/canonicals completed; OG/Twitter fields generated; baseline JSON-LD added; sitemap/robots/llms.txt created or fixed; security headers prepared via the .htaccess recipe; defer applied to render-blocking external scripts; dimensionless images flagged. The security boundary holds: SafeFix never adds CSP-breaking tricks such as inline-onload, and never produces a duplicate policy when the site has its own CSP.

When SafeFix is not done is also defined: bulk edits in third-party theme files of unclear origin, text interventions that change meaning, risky rewrites dependent on server behaviour, and brand/language changes requiring client approval. These items go to human approval or external action.

SafeFix is not a live production sign-off. The output package must be uploaded, caches cleared and the effect confirmed via Deploy-Verify with independent tools.

- fixed ZIP
- changed files list
- rollback manifest
- before/after internal score
- external action list
- deploy verification checklist

What the engine does: Applies low-risk reversible fixes on a working copy.

What it doesn't: No rebuilds; no approval-needing content edits; no live sign-off.

What the output is: fixed ZIP + changed files + rollback + before/after score.

How it ports to the website: The SafeFix page carries the 'fix-but-reversible' promise and the proof list.

Meaning for the executive: Controlled risk closure with a guaranteed way back.

Meaning for the technical team: Every change is traceable; diff + manifest give disciplined delivery.

8. Build Mode — Deep Dive

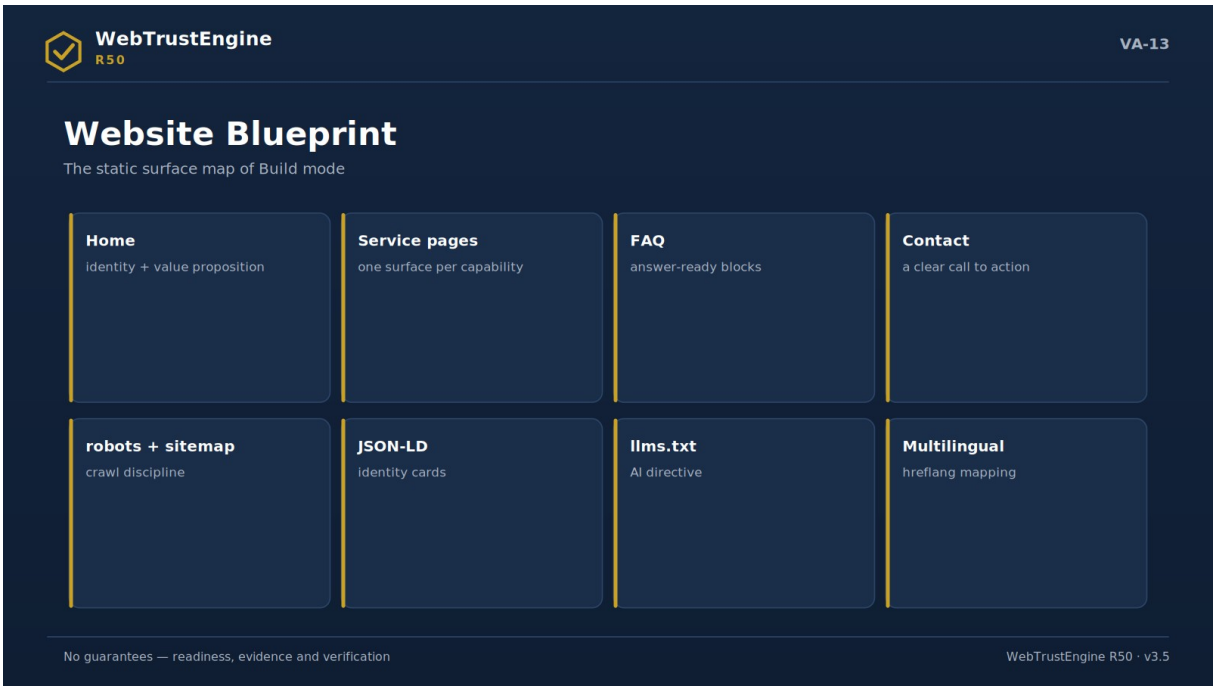

Figure VA-13 — The website blueprint: eight building blocks of the static surface.

Build is for producing new pages or sites; yet public copy deliberately avoids the claim 'builds and takes live'. Build produces consistent static surfaces by component, sector, language, structured data, SEO and AEO/GEO rules. Production follows the claim-safe content principle: no guarantee language, no unverifiable claims.

A static architecture is preferred: few dependencies, high auditability, easy portability. Schema-ready sections (Organization/Service/FAQ) are built in from the start; multilingual structure is set with hreflang

and consistent identity signals. Build output is not a publish decision; pre-publish human approval and deploy verification are required.

- component/sector/language rules
- claim-safe content
- schema-ready sections
- multilingual consistency
- publish decision separate

What the engine does: Produces consistent, schema-ready static surfaces.

What it doesn't: No auto go-live; no guarantee language.

What the output is: A publish-candidate package + approval checklist.

How it ports to the website: The Build page frames 'it produces, you decide to publish'.

Meaning for the executive: Fast, auditable production capacity.

Meaning for the technical team: Component rules give reusability.

9. Monitor / Deploy-Verify — Deep Dive

Figure VA-23 — The Deploy-Verify loop: seven steps from upload to the diff report.

Figure VA-30 — The evidence chain: six links from the baseline record to the diff report.

File produced ≠ working live. This inequality is the most commonly skipped gap in web work, and one of WebTrustEngine's strongest differentiators stands exactly here. An .htaccess being in the package does not mean hosting reads it; an OG tag being in the HTML does not mean the social platform cache sees it.

The intervening layers are enumerable and predictable: hosting cache, CDN cache, whether .htaccess is actually processed, social-preview caches, browser cache. Deploy-Verify turns each into an evidence step: live header fetch, redirect tracing, metadata/schema comparison, sitemap/robots/llms.txt reachability, social-preview debugger checks.

The external-tool layer is part of the loop: SecurityHeaders and Observatory grade headers; SSL Labs evaluates certificate/protocol; Lighthouse/PageSpeed produce lab metrics; Search Console and Bing Webmaster show indexing. The engine does not produce these results; it bridges and binds them to the report.

The typical post-GoDaddy flow: (1) files uploaded; (2) hosting/CDN caches purged; (3) live headers fetched and compared with the expected set; (4) redirects and 404 behaviour tested; (5) social-preview debuggers refreshed; (6) independent tool passes run; (7) the Deploy-Verify report produced a diff table against baseline. This report turns 'the package was produced' into 'the live effect is verified'. It is not a real-time SOC; it is a periodic verification loop.

- hosting/CDN cache
- live header fetch
- social preview debuggers
- Search Console / Bing
- Lighthouse / PageSpeed
- SecurityHeaders / Observatory / SSL Labs

What the engine does: Compares live against baseline; reports diffs.

What it doesn't: Not real-time monitoring (SOC); does not produce scores itself.

What the output is: Deploy-Verify diff report + external-tool bridge results.

How it ports to the website: The Deploy-Verify page is built as the product's strongest story.

Meaning for the executive: Closing the gap between 'we did it' and 'it works'.

Meaning for the technical team: Cache layers are managed by procedure; surprises drop.

10. Evidence Package

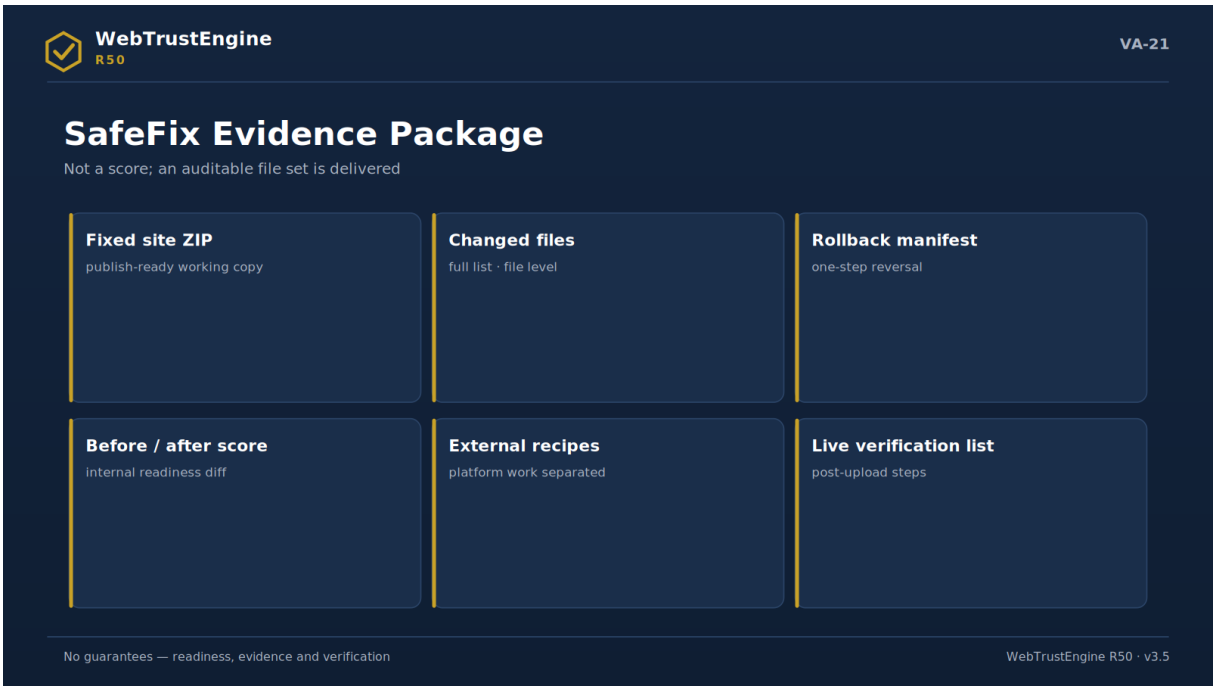

Figure VA-21 — The SafeFix evidence package: six delivery items.

Every WebTrustEngine run leaves an evidence package; the report is not a PDF page but an auditable file set. Its core: before score, after score, changed files, rollback manifest, external actions, the runtime-checks list, manual-verification items, the QA report, the manifest and checksums.

This structure delivers two things at once: the story for the executive (what changed, what was gained) and verifiability for the auditor (which file, which hash, which boundary). Checksums are computed from real files after packaging; the manifest lists every file's size and type. The evidence package makes 'trust us' unnecessary — the files speak.

- before/after score
- changed files
- rollback manifest
- external actions
- runtime checks
- manual verification
- QA report
- manifest + checksum

What the engine does: Produces an auditable file set per run.

What it doesn't: Does not produce a one-page 'trust us' report.

What the output is: A 10-part evidence set (score→checksums).

How it ports to the website: 'Your output is an evidence package, not a report' goes on the homepage.

Meaning for the executive: Auditability equals credibility.

Meaning for the technical team: Hash/manifest discipline locks delivery quality.

11. Security Boundary

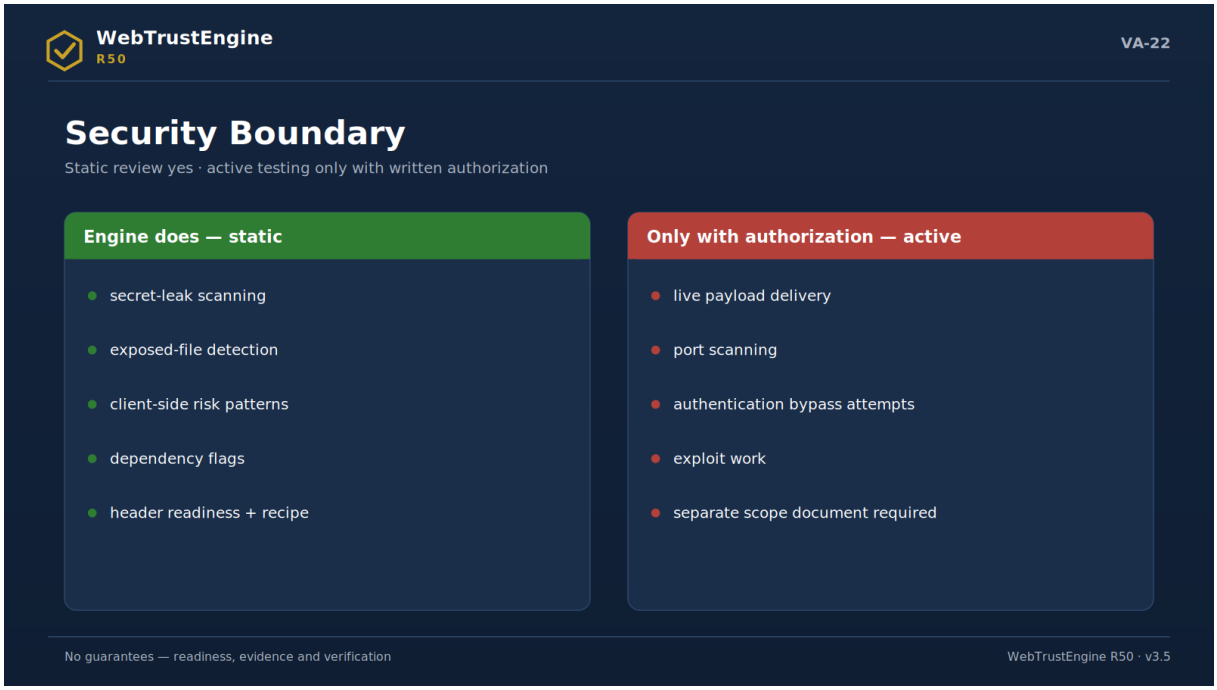

Figure VA-22 — The security boundary: static review is engine work; active testing needs separate authorization and scope.

A static security review is security analysis over source, configuration and output files without touching the target system. WebTrustEngine's security layer stays within this definition and has four blocks: secret scan (key/password/token leakage), exposed files (.env, backups, .git, config exposure), client-side risky patterns (eval, document.write, dangerous sinks) and server-side SAST classifications (SQLi/command/traversal/deserialization/SSRF/XXE markers).

SCA/CVE logic accompanies these: risky ranges of known library versions are flagged; but no 'this version is exploitable' claim is made — the flag binds to an update recipe. CSP/HSTS readiness and OWASP mapping put findings into a shared language.

When is it not a pentest? Always — in this product. No active payloads, no live port scans, no authentication-bypass attempts, no exploit generation. Any DAST-class behaviour requires written authorization, ownership verification and a separate scope document. This legal and ethical boundary is not a weakness; it is trust discipline: the client knows exactly what they bought and what must be ordered separately.

- secret scan

- exposed files
- JS sink patterns
- SCA/CVE flags
- CSP/HSTS readiness
- OWASP mapping
- DAST = authorization + scope

What the engine does: Runs a static security review over source.

What it doesn't: No exploitation/port scans/payloads; not a pentest.

What the output is: Risky-pattern report + recipes + OWASP mapping.

How it ports to the website: The Security page shows 'does / doesn't' in two columns.

Meaning for the executive: A visible dashboard of silent risks.

Meaning for the technical team: The SAST ↔ DAST line holds in contract language too.

12. AI / GEO / AEO Readiness — Deep Dive

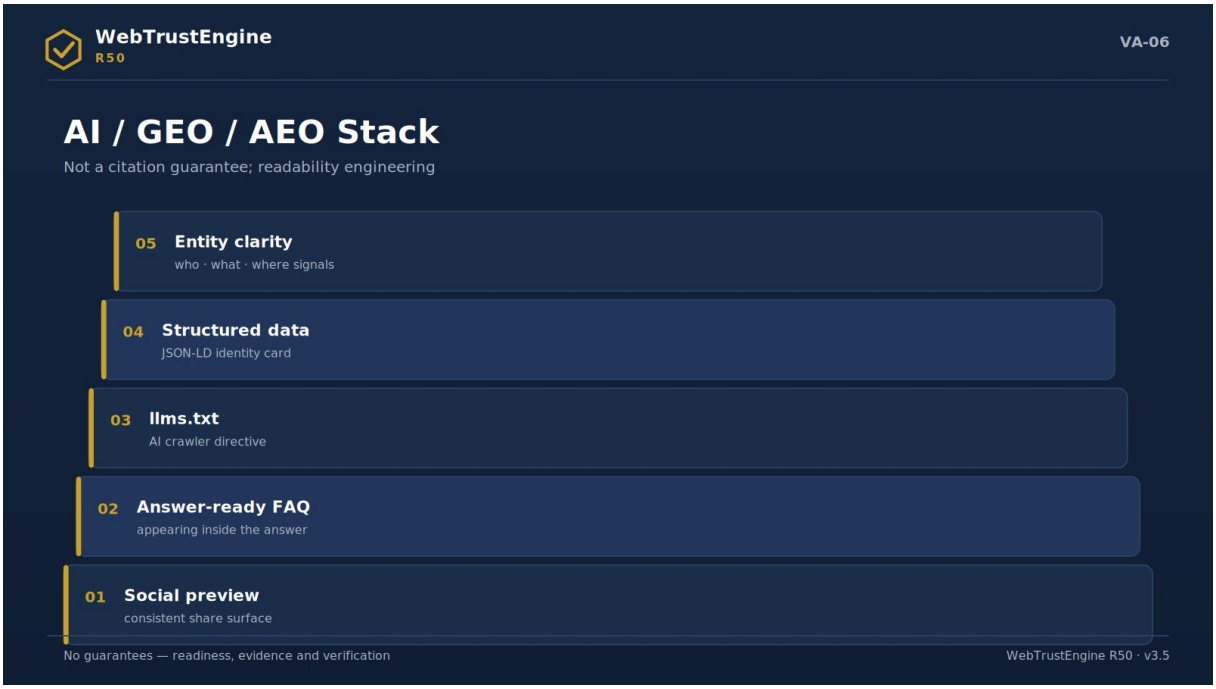

Figure VA-06 — The AI/GEO/AEO readiness stack: five levels from identity to answer-ready content.

AI systems do not read the web like humans; they look for identity, structure and sourceability. If an answer engine cannot quickly and consistently resolve 'who runs this site, what service is offered, which page answers what', the content will not enter the answer no matter how good it is. AI/GEO/AEO readiness is therefore not a trend but readability engineering.

The building blocks: entity clarity (Organization/Person identity, the sameAs chain), service clarity (naming and defining the service), answer-ready FAQ (Q&A blocks and schema), structured-data consistency, llms.txt and AI-crawler directives, canonical/hreflang discipline, social-preview integrity and topical consistency across pages.

Public claim-safety applies here too: never 'AI recommends us'; always 'readability readiness for AI'. There is no citation guarantee; there are readiness signals — measurable, auditable, improvable.

- entity clarity
- service clarity
- answer-ready FAQ
- structured data
- llms.txt
- canonical/hreflang
- social preview
- topical consistency
- no citation guarantee

What the engine does: Prepares and marks readability signals.

What it doesn't: Gives no citation/recommendation guarantee.

What the output is: AI-readiness finding set + llms.txt suggestion.

How it ports to the website: The AI/GEO page frames 'readiness — not a guarantee'.

Meaning for the executive: Visibility readiness in the answer era.

Meaning for the technical team: Signals are auditable; measured, not promised.

13. Structured Data & Entity Clarity — Deep Dive

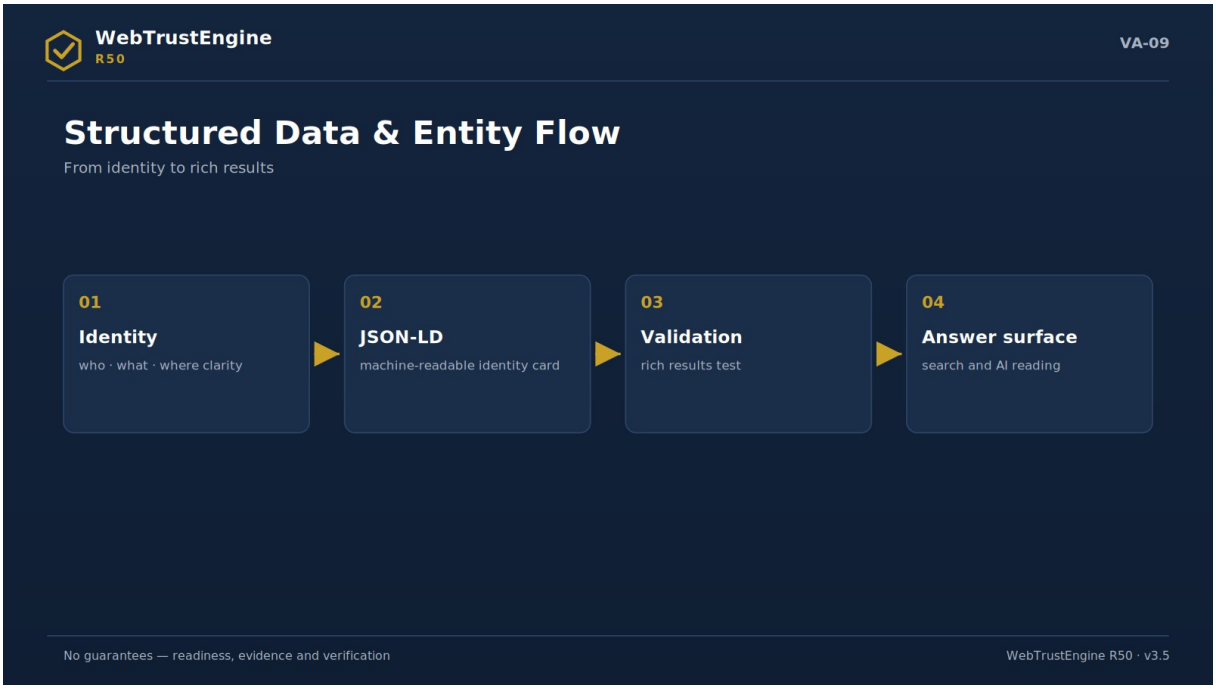

Figure VA-09 — The entity flow: from identity clarity to machine-readable identity and a validated answer surface.

Used correctly, the schema layer is the brand's machine identity; used wrongly, it is invisible debt. Organization and WebSite establish site identity; WebPage gives page context; Service names the offering;

FAQPage opens Q&A to answer engines; BreadcrumbList explains hierarchy; Person binds founder/expert identity.

Types like Product, Event, Recipe are conditional: used only when that content truly exists. Wrong-schema risk is real — an unsuitable type or missing required field can break rich-result eligibility and invert the trust signal. The engine parses JSON-LD, audits type-page fit and field integrity; for entity consistency it checks cross-page coherence of signals such as sameAs/phone/address.

- Organization/WebSite/WebPage
- Service/FAQPage/BreadcrumbList
- Person
- Product/Event = conditional
- JSON-LD parse + field integrity
- entity consistency

What the engine does: Parses schema, audits fit and integrity.

What it doesn't: Never invents schema for non-existent content.

What the output is: Type-based findings + baseline block generation.

How it ports to the website: The Structured Data page carries a type table + conditional-rule note.

Meaning for the executive: The health of machine identity.

Meaning for the technical team: Conditional rules prevent wrong-schema debt.

14. Accessibility Readiness — Deep Dive

Figure VA-10 — The accessibility path: from automated detection to manual verification.

Accessibility is governed in two layers. The static layer belongs to the engine: alt presence and quality markers, form-label matching, heading order (H1→H2→H3), landmark structure

(header/nav/main/footer), skip link, language attribute, table headers. This layer is fast, repeatable and objective.

The second layer is human and deliberately not automated: the context of a contrast decision, the real look of focus visibility, the meaning of reading order, the adequacy of media description. These WCAG items demand visual/auditory judgment; hence the manual-verification rule, and 'full WCAG compliance guarantee' appears in no text. The correct promise: static readiness + flagged manual items + clear routing to a reviewer.

- alt/label/heading/landmark/skip link
- language attribute + table semantics
- contrast readiness = manual
- no WCAG guarantee

What the engine does: Detects and fixes static accessibility signals.

What it doesn't: Never auto-PASSes items needing render/human judgment.

What the output is: A11y finding set + manual-item list.

How it ports to the website: The Accessibility page tells the two-layer story.

Meaning for the executive: Inclusion plus legal-risk visibility.

Meaning for the technical team: Manual items bind to the reviewer flow.

15. Performance Readiness — Deep Dive

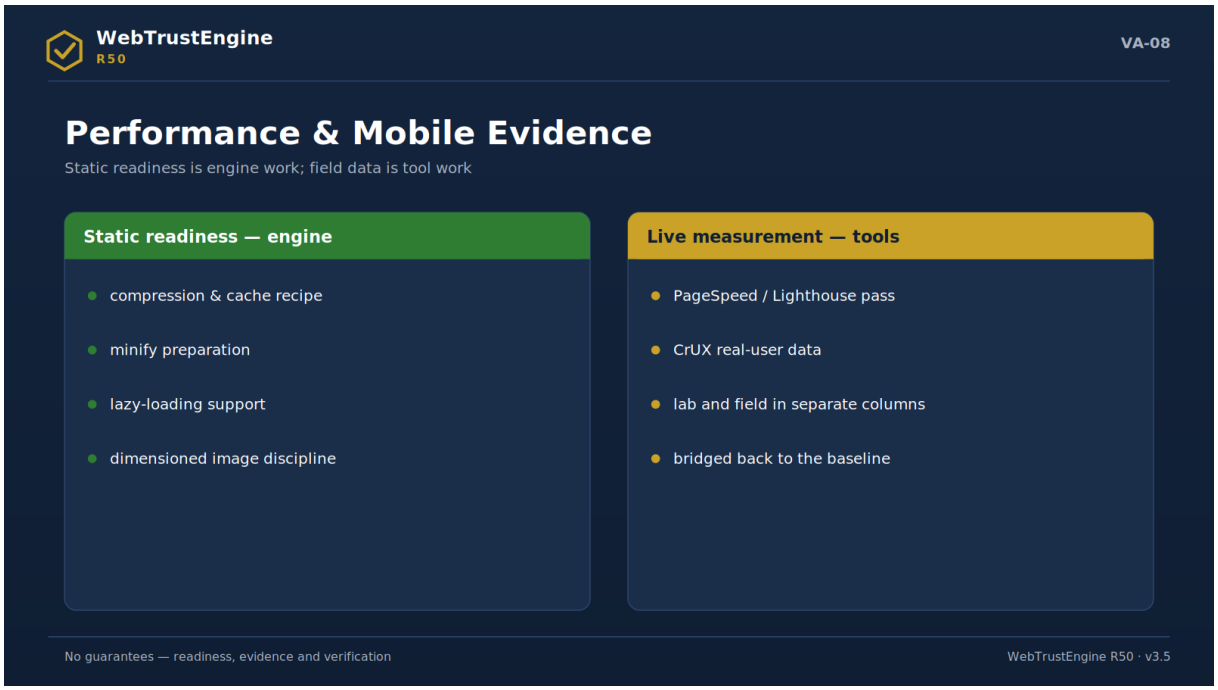

Figure VA-08 — Performance evidence on two shores: static readiness in the engine, live measurement in independent tools.

Static performance readiness means removing in-file blockers that cap a page's speed potential. The engine's territory: a compression/cache recipe (.htaccess), render-blocking script/CSS detection with defer suggestions, image format and dimension signals, lazy-loading markers, CSS/JS minify readiness.

What is not the engine's territory is written just as clearly: producing real field CWV, measuring user data, configuring the CDN itself, changing hosting resources. The first two are runtime bridges (PageSpeed/CrUX), the last two external actions (Cloudflare/hosting recipes). This triple split keeps performance talk from ever inflating into 'the engine made it fast'; it stays on the chain 'the engine prepared, live measurement verified, the infrastructure recipe was applied'.

- compression/cache recipe
- render-blocking + defer
- image format/lazy
- minify readiness
- CDN = external action
- CWV = not field measurement

What the engine does: Detects and prepares in-file speed blockers.

What it doesn't: Produces no field CWV; doesn't change CDN/hosting itself.

What the output is: Perf readiness findings + the .htaccess recipe.

How it ports to the website: The Performance page shows three columns: readiness/measurement/infrastructure.

Meaning for the executive: Correct ordering of speed investment.

Meaning for the technical team: The lab/field/infra split preserves measurement discipline.

16. Privacy / Cookie Readiness — Deep Dive

Privacy readiness starts with visibility: which trackers exist on the page, what runs before consent, which flags cookies carry, where the policy links are. This is the static counterpart of the Blacklight/Cookiebot approach, and the engine produces these signals.

The boundary sits in the same paragraph: the engine gives no legal-compliance guarantee; it is not KVKK/GDPR counsel. Visibility and readiness come from the engine; the compliance judgment comes from the legal layer. Public copy carries this duality in one sentence: 'tracker visibility and cookie readiness — not legal advice'.

- tracker visibility
- cookie banner readiness
- pre-consent markers
- no legal guarantee

What the engine does: Makes tracker/cookie signals visible.

What it doesn't: Makes no legal-compliance judgment.

What the output is: Privacy finding set + policy-link suggestion.

How it ports to the website: The Privacy section uses the 'visibility + readiness + legal sign-off' chain.

Meaning for the executive: Early visibility of reputation risk.

Meaning for the technical team: Consent flow is a product decision; the engine supplies signals.

17. External Action Recipes — Deep Dive

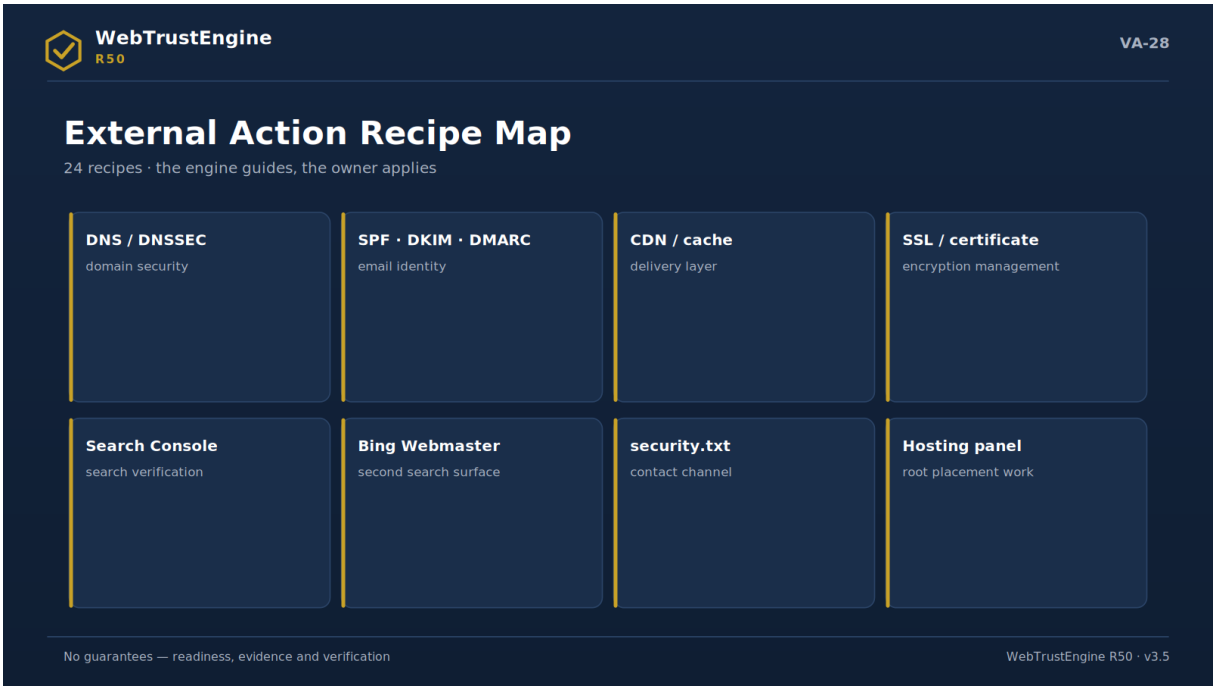

Figure VA-28 — The external action map: eight platform areas the engine cannot change in files.

An external action recipe turns steps the engine cannot perform in-file — yet which directly affect web quality — into manageable instructions. The engine never marks these as 'auto-fixed'; it produces recipes answering who, where and in what order.

The recipe universe groups into 24 headings with typical examples: enabling DNSSEC; the SPF/DKIM/DMARC email identity chain; MX layout; Cloudflare security level and WAF rules; HSTS preload submission; Search Console and Bing Webmaster verification; IndexNow; CDN cache purge; hosting server headers; TLS server configuration; security.txt placement; social-preview debugger refresh.

Every recipe follows the same template: where it applies (panel/record/service), concrete steps, verification path. 'DNS work' stops being a vague task and becomes an actionable item the owner can execute and the engine can later verify.

- DNSSEC · SPF/DKIM/DMARC · MX
- Cloudflare · WAF · HSTS preload
- Search Console · Bing · IndexNow
- CDN purge · hosting headers · TLS config
- security.txt · preview debugger

What the engine does: Turns external steps into actionable recipes.

What it doesn't: Doesn't perform panel/DNS work itself, never claims it did.

What the output is: A 24-heading recipe set (steps + verification).

How it ports to the website: 'Engine + recipe' framing on Tool Ecosystem/Service pages.

Meaning for the executive: A clear ownership list of external dependencies.

Meaning for the technical team: The step+verify template cuts operational error.

18. Runtime Bridges — Deep Dive

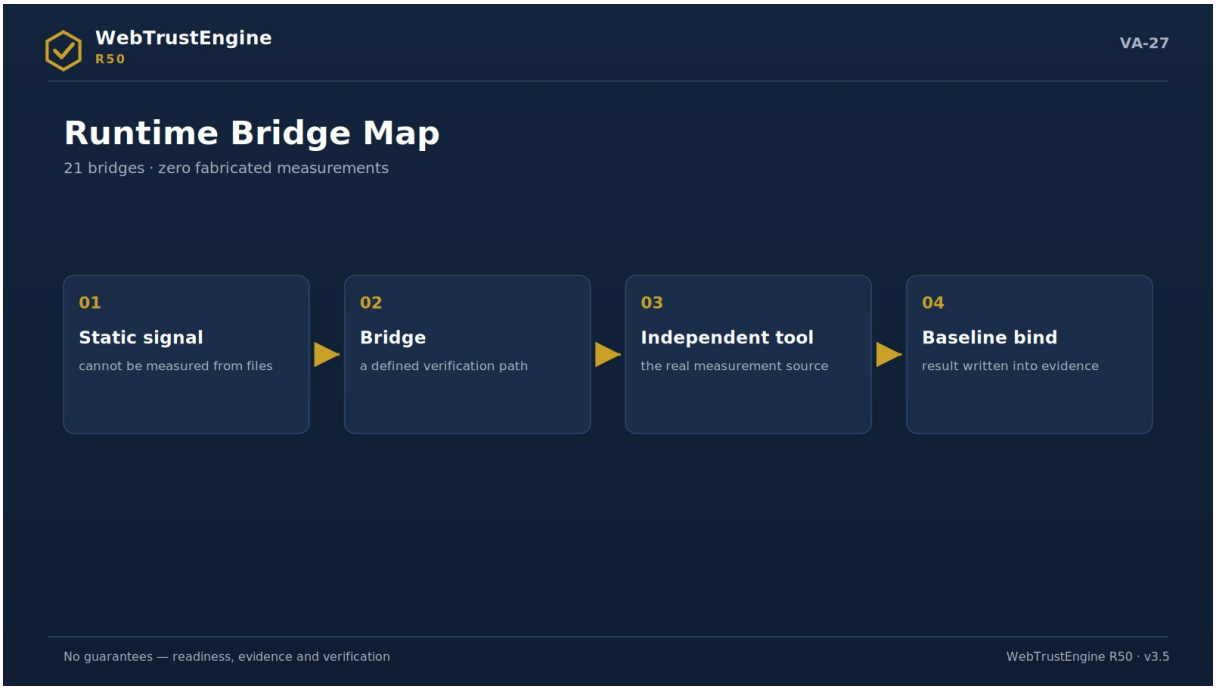

Figure VA-27 — The runtime bridge: four steps from static signal to independent measurement.

A runtime bridge is the bridge layer for checks that cannot be exactly measured from a static file. The principle is simple: the engine fabricates no measurements. Offline, the bridge says 'requires_live' and writes which tool measures which metric and how; live, it routes to the real measurement.

The bridge universe groups into 21 checks with typical examples: SecurityHeaders (header grade), Mozilla Observatory, SSL Labs (TLS grade), PageSpeed/Lighthouse (lab metrics), CrUX (field CWV), social-preview inspectors, redirect-chain tracing, uptime/response headers, the contrast runtime check, a link checker for broken external links.

This layer answers two questions at once: 'who produces live scores' (independent tools) and 'why is the engine still needed' (because the bridge binds measurement to baseline and report; a lone tool score is not a decision surface).

- SecurityHeaders · Observatory · SSL Labs
- PageSpeed · Lighthouse · CrUX
- preview inspectors · redirect chain
- uptime · response headers · contrast · link checker

What the engine does: Binds live measurements to real tools.

What it doesn't: Fabricates no scores; replaces no tools.

What the output is: The bridge list + tool/metric/how template.

How it ports to the website: A bridge map in the 'independent verification' section.

Meaning for the executive: The source of scores is transparent.

Meaning for the technical team: Baseline-bound measurement fixes lone-score context loss.

19. Manual Verification — Deep Dive

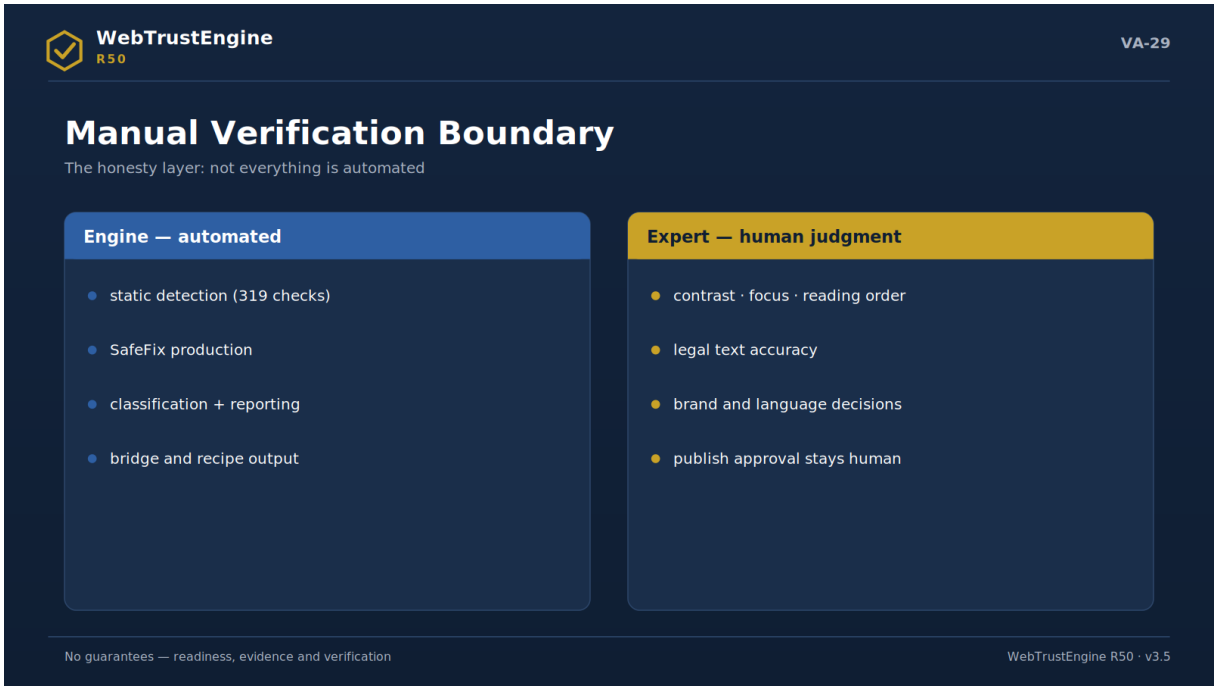

Figure VA-29 — The manual verification boundary: the engine's automated area versus human judgment.

Manual verification is the engine's honesty layer: some areas cannot be automated and are not presented as if they were. Legal claims (compliance statements, copyright text), certain accessibility decisions (contrast context, adequacy of media description), cultural context and brand voice, content accuracy (figures, titles, claims), changes requiring client approval, and steps requiring security authorization live here.

Its operation is clear: the item is flagged, the rationale written, and it is routed to a reviewer/decision-maker; the engine never auto-PASSes these items. This design choice also has marketing value: against products claiming 'we automated everything', the product that states plainly what requires human judgment is more credible to an enterprise buyer.

- legal · accessibility context
- cultural context · brand voice
- reality check · client approval
- security authorization

What the engine does: Flags and routes items requiring human judgment.

What it doesn't: Never auto-PASSes these items.

What the output is: Manual-item list + rationale + routing.

How it ports to the website: The 'honesty layer' story goes to FAQ and Claim-Safety pages.

Meaning for the executive: The credible limit of automation claims.

Meaning for the technical team: The reviewer flow is a formal part of the process.

20. Public Claim-Safety Matrix

Figure VA-14 — The claim-safe language matrix: prohibited and preferred language side by side.

Figure VA-25 — The public claim matrix: what is never promised and what is proven.

The claim-safety matrix is not just a marketing filter; it is the product's legal and ethical defence line. The left column shows what may be said, the middle what must not be, the right column why. The rule is simple: no unverifiable result is promised; readiness, evidence and the verification chain are.

✓ Safe phrase	✗ Risky phrase	Why?

readiness	guaranteed compliance	Guarantee claim
verified post-deploy	passed live	Needs live testing
static security review	full pentest	Needs authorization + live testing
reference catalog	automated checks	Number-inflation risk
SafeFix package	production-ready	Publish decision is separate
before/after internal score	full marks on all tools	Independent-tool result claim
AI-readability readiness	AI recommends us	No citation guarantee

21. Website Content System

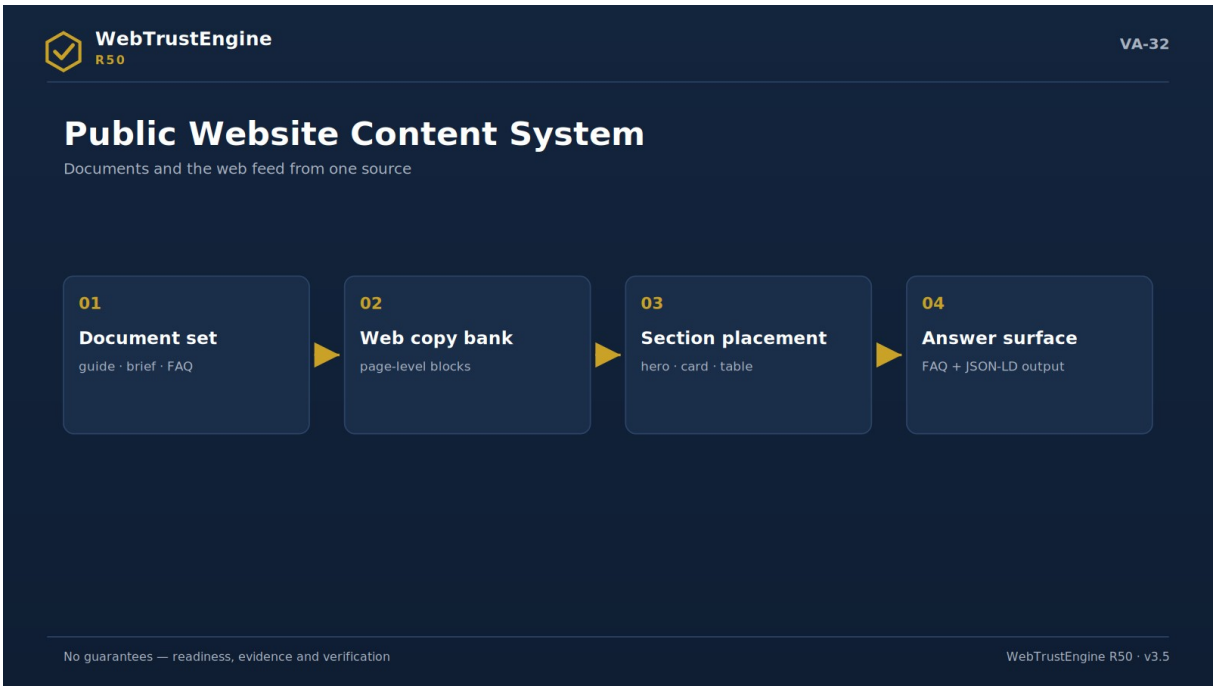

Figure VA-32 — The content system: one line from documents to web sections and the answer surface.

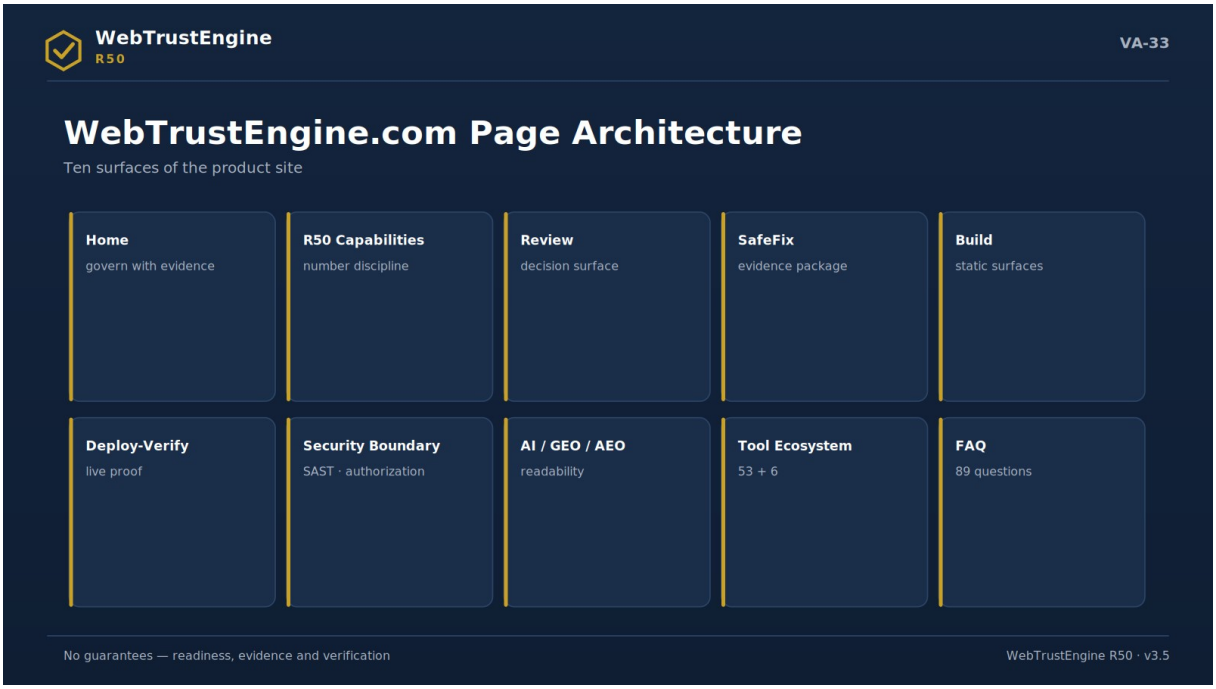

Figure VA-33 — The page architecture: ten product surfaces from Home to FAQ.

The page copy below ports directly to both sites; each page ships meta, hero, CTA, five sections, five FAQs and a suggested visual.

webtrustengine.com

Home

Meta title: WebTrustEngine — Evidence-First Web Governance Engine

Meta description: Evaluate your web presence across 10 domains with evidence: security, SEO, structured data, accessibility, AI/GEO/AEO. 319 implemented checks, no open roadmap.

Hero: Evidence-first web governance engine — Review, safely fix, build and verify live.

CTA: Run a Review · See R50 Capabilities · Explore the Evidence Package

Sections:

- Why governance? — The web is a multi-audit surface; a repeatable evidence loop beats one-off fixes.
- Four modes — Review as decision surface, SafeFix as safe repair, Build as production, Monitor as live verification.
- Ten domains — One before/after readiness frame from security to AI readability.
- Evidence package — Not a score but a file set: changed files, rollback, recipes, checksums.
- Honest boundaries — No guarantees; readiness, evidence and independent verification.

FAQ:

- Is this a scanner? → No; a governance engine running an evidence loop.
- 2,033 checks? → No; a reference catalog. Working checks: 319.
- A pentest? → No; a static security review.
- Ranking guarantee? → No; it produces readiness.
- Who gives live scores? → Independent tools post-deploy.

Suggested visual: VA-17

Implementation note: Four number cards (319/2,033/26/0) under the hero; meta title within 60 chars.

R50 Capabilities

Meta title: R50 Capabilities — 319 Implemented Checks, No Open Roadmap

Meta description: The R50 number contract: 2,033 reference catalog, 319 implemented checks, 68 security patterns, 26 SafeFix, 21 runtime bridges, 24 external recipes.

Hero: The R50 capability map — Every number defined; none inflated.

CTA: See the Number Ledger · Explore Catalog Distribution

Sections:

- Reference catalog — A 2,033-signal map — not a check count.
- Working layer — 319 detection checks: 80 core + 239 granular.
- Security patterns — 68 static patterns; classification, not exploitation.
- Bridges and recipes — 21 runtime bridges + 24 external-action recipes.
- ROADMAP = 0 — Every item in its correct artifact class.

FAQ:

- Where does 319 come from? → 80 core + 239 granular; CLI/gate/registry agree.
- What is ENTEGRE 1,258? → Catalog row coverage; not a check count.
- What ROADMAP=0 is not? → Not 'everything automated'.
- Are numbers auditable? → Yes; row by row in the registry and matrices.
- Why two levels? → Honesty: the map is not the engine.

Suggested visual: VA-18

Implementation note: Each number card should link to the number ledger (App. 24).

Review

Meta title: Review — The Evidence Baseline

Meta description: ZIP, local folder or authorized live URL: a risk map and decision surface with zero file changes.

Hero: Snapshot first, decision next — Review changes nothing; it classifies and scores.

CTA: Run a Review · See a Sample Report

Sections:

- Three input types — ZIP for delivery audit, local folder for iteration, live URL only with authorization.
- Six decision questions — Where is risk, what closes via files, what needs external platforms, what needs live verification, what needs approval.
- Score table — 10 domains, internal readiness out of 100.
- Findings list — Classified by domain/severity/file.
- Next step — Suggested SafeFix scope + external-action list.

FAQ:

- Does it touch files? → No.
- Does it hit live? → Only with ownership/authorization.
- Pass/fail? → No; a context-owned decision surface.
- How long? → Depends on site size; output structure is constant.
- Report format? → Score + findings + class distribution + suggestion.

Suggested visual: VA-02

Implementation note: Three input types as tabs; keep 'no file changes' in the hero.

SafeFix

Meta title: SafeFix — The Safe, Reversible Fix Package

Meta description: Low-risk fixes on a working copy: changed-file list, rollback manifest, before/after score.

Hero: Fix — but reversible — No rebuilds; it closes gaps while preserving value.

CTA: See SafeFix Scope · How Rollback Works

Sections:

- Philosophy — Safe improvement, not rebuilding.
- Scope — Meta/OG/JSON-LD, sitemap/robots/llms.txt, header recipe, static performance.
- Three proofs — Changed files, rollback manifest, before/after score.
- Security boundary — No CSP-breaking tricks; no duplicate policies.
- When it is skipped — Approval-needing content, risky rewrites, unclear theme files.

FAQ:

- If the site breaks? → The rollback manifest reverts in one step.
- Live sign-off? → No; upload + Deploy-Verify required.
- What does it fix? → Meta layer, delivery hygiene, header readiness.
- Does it touch content? → No meaning-changing edits.
- Who approves? → Publishing is always a human decision.

Suggested visual: VA-21

Implementation note: Rollback emphasis before the visual; 'reversible' near the H1.

Build

Meta title: Build — Schema-Ready Static Production

Meta description: Consistent static surfaces by component, sector, language and claim-safety rules; you decide to publish.

Hero: It produces — you decide to publish — Static architecture: few dependencies, high auditability.

CTA: See Build Rules

Sections:

- Static architecture — Portable, auditable, fast.
- Schema-ready sections — Organization/Service/FAQ from the start.
- Multilingual — hreflang + consistent identity signals.
- Claim-safe content — No guarantee language; no unverifiable claims.
- Approval flow — Pre-publish human approval + deploy verification.

FAQ:

- Does it go live? → No; it produces a candidate package.
- By which rules? → Component/sector/language/schema/claim-safety.
- Does it break the site? → Build produces a separate surface.
- SEO-fit? → Technical readiness rules are built in.
- Who publishes? → You; with an approval checklist.

Suggested visual: VA-13

Implementation note: Repeat the 'you decide to publish' frame right above the CTA.

Monitor / Deploy-Verify

Meta title: Deploy-Verify — Producing Files Is Not Enough

Meta description: After upload, live headers, redirects, previews and external-tool results are confirmed with evidence.

Hero: Deploy. Verify. Prove. — 'Package produced' → 'live effect verified'.

CTA: See the Verification Flow

Sections:

- Cache layers — Hosting, CDN, social preview, browser — all in the procedure.
- Live fetch — Headers compared against the expected set.
- External-tool pass — SecurityHeaders/SSL Labs/PageSpeed bound to the report via bridges.
- Diff table — Baseline ↔ live diff report.
- Periodic loop — Not a SOC; an evidence-verification rhythm.

FAQ:

- Why needed? → A file existing doesn't prove live effect.
- Who measures? → Independent tools; the engine bridges.
- After GoDaddy? → A 7-step flow: upload→purge→fetch→test→debug→tools→report.
- Real-time? → No; periodic.
- Output? → Diff report + tool-result bridges.

Suggested visual: VA-23

Implementation note: The 7-step flow as a numbered list; a diff-report mock via VA-30.

Security Boundary

Meta title: Security Boundary — Not a Pentest

Meta description: Static security review: secrets/exposure/risky patterns/SAST + OWASP mapping. Active testing only with authorization.

Hero: SAST yes — DAST only with authorization — A clear boundary is the source of trust.

CTA: See the Security Layer

Sections:

- Static layer — Secret scan, exposed files, JS sink patterns, server SAST.
- SCA/CVE — Risky-version flags → update recipes.
- Header readiness — CSP/HSTS recipes + OWASP mapping.
- Forbidden behaviour — No payloads, no port scans, no bypass.
- Authorization model — DAST = written authorization + ownership + scope.

FAQ:

- A pentest? → No.
- Exploit code? → Never.
- OWASP? → Maps findings to a shared language.
- If active testing is wanted? → A separate authorized scope.
- Why the boundary? → Law + ethics + credibility.

Suggested visual: VA-22

Implementation note: 'Does/Doesn't' as a two-column table; DAST terms in a box.

AI / GEO / AEO

Meta title: AI / GEO / AEO — Readiness for the Answer Era

Meta description: Entity clarity, structured data, llms.txt, answer-ready FAQ: AI readability — not a citation guarantee.

Hero: Readiness to appear inside the answer — AI reads the web via identity, structure and sourceability.

CTA: See the AI Readiness Signals

Sections:

- Entity clarity — Who/what/where — with the sameAs chain.
- Answer-ready FAQ — Q&A blocks + schema.
- llms.txt — AI-crawler directives.
- Consistency — canonical/hreflang + topical consistency.
- Boundary — No citation guarantee; readiness instead.

FAQ:

- Will AI recommend us? → Not guaranteed; readability is prepared.
- What is llms.txt for? → AI access/summary directives.
- Is schema required? → Critical for identity clarity.
- Multilingual sites? → Via hreflang discipline.
- Measurable? → The signals are auditable.

Suggested visual: VA-06

Implementation note: 'Not a guarantee' on the hero sub-line; llms.txt sample in a code block.

Tool Ecosystem

Meta title: Tool Ecosystem — 53 Tools + 6 Standards

Meta description: From SecurityHeaders to PageSpeed, OWASP to WCAG: the engine doesn't replace tools, it governs their signals.

Hero: Readiness verified by independent tools — A base of 53 canonical tools + 6 standards/frameworks.

CTA: See the Ecosystem Map

Sections:

- Security — SecurityHeaders · Observatory · SSL Labs.
- Performance — PageSpeed · Lighthouse · CrUX.
- SEO — Search Console · Bing · Screaming Frog.
- Accessibility + data — axe-core · Rich Results · Schema.org.
- Standards — OWASP · WCAG 2.2 · CWV · W3C · llms.txt.

FAQ:

- Instead of tools? → No; bridging and classification.
- Who produces scores? → The tools themselves.
- Why 53+6? → A canonical base; not an arbitrary list.
- Do results enter the report? → Bound to baseline via bridges.
- Can tools be added? → The ecosystem grows by release.

Suggested visual: VA-15

Implementation note: Text list instead of tool logos (no brand permissions needed); bridge note below.

FAQ

Meta title: FAQ — Honest Answers

Meta description: What 2,033 means, is it a pentest, an SEO guarantee, how live scores verify — all clear.

Hero: The honest answer to the question — The most critical questions in claim-safe language.

CTA: See the Full FAQ

Sections:

- Numbers — 319 vs 2,033 split.
- Security — The SAST ↔ DAST line.
- Scores — Internal readiness vs independent tools.
- Boundaries — What we do not guarantee.
- Process — Review→SafeFix→Deploy-Verify→Monitor.

FAQ:

- Most asked? → 'Is 2,033 automated?' — No, a reference catalog.
- Second? → 'A pentest?' — No, a static review.
- Third? → 'Guarantees?' — Readiness and evidence.
- Fourth? → 'Scores from whom?' — Independent tools.
- Fifth? → 'If it breaks?' — Revert via rollback.

Suggested visual: VA-25

Implementation note: Top 5 questions above the fold; one link to the full FAQ.

Contact / Run Review

Meta title: Contact — Run a Review

Meta description: Start a Review for an evidence snapshot of your web presence; let's talk scope components.

Hero: First step: the evidence snapshot — We talk scope, not price: size, languages, modes, monitoring.

CTA: Run a Review · Schedule a Scoping Call

Sections:

- Input — ZIP/folder/authorized URL.
- Output — Score + findings + classes + suggestion.
- Process — Review→decision→SafeFix→verification.
- Scope — Size/languages/modes/monitoring cadence.
- Boundary — No pricing on this page.

FAQ:

- How do we start? → With a Review input.
- How long? → Size-dependent; structure constant.
- Confidentiality? → Input used only for evaluation.
- Then? → Together on the decision surface.
- Price? → Discussed via scope components.

Suggested visual: VA-24

Implementation note: Form fields: input type + size + languages only; no price field.

vatanseverbilisim.com

WebTrustEngine Hizmeti

Meta title: WebTrustEngine Service — Engine + Expert Management

Meta description: Vatansever Bilişim delivers the R50 engine with expert judgment and deployment support.

Hero: Engine + human judgment + deployment support — Not just software; a managed evidence loop.

CTA: Explore the Service · Talk to an Expert

Sections:

- Consulting — Baseline assessment, priority map, evidence interpretation.
- Managed service — Running Review→SafeFix→Deploy-Verify→Monitor.
- Deploy support — GoDaddy/Cloudflare upload + live verification.
- Expert layer — Manual-verification items closed by humans.
- Boundary — No pricing; no guarantee language.

FAQ:

- What's different? → Engine + expert together.
- Who executes? → External actions by expert/owner.
- Reports for whom? → Executive language + a technical annex.
- Continuity? → Via the Monitor rhythm.
- Price? → Discussed by scope.

Suggested visual: VA-11

Implementation note: The 'engine + expert' split on the first screen; link to the role table (S27).

Corporate Web Governance

Meta title: Corporate Web Governance — A Measurable Web Asset

Meta description: Govern your site across 10 domains with evidence: scores, changed files, rollback, verification.

Hero: From storefront to asset — A web asset is measured, improved, evidenced.

CTA: See the Approach

Sections:

- Problem — Scattered scores are not a decision surface.
- Frame — 10 domains in one report.
- Evidence — A file set + checksums.
- Verification — Post-deploy with independent tools.
- Rhythm — A periodic governance loop.

FAQ:

- One-off? → No; a loop.
- Report language? → Executive + technical.
- Existing agency? → Complements as the evidence layer.
- Start? → With Review.
- Guarantees? → Readiness and evidence.

Suggested visual: VA-01

Implementation note: The 10-domain visual (VA-19) beside the text; 'storefront→asset' in the hero.

Technical SEO and Security Readiness

Meta title: Technical SEO and Security Readiness

Meta description: Header recipes, CSP/HSTS readiness, meta/canonical/sitemap discipline — one package.

Hero: Trust and findability together — Security headers + technical SEO in one evidence chain.

CTA: See the Scope

Sections:

- Headers — HSTS/CSP/nosniff recipes.
- SEO base — Meta/canonical/robots/sitemap.
- Conflict control — Double-CSP and inline conflicts.
- Verification — SecurityHeaders + Search Console.
- Boundary — No ranking guarantee.

FAQ:

- A+ guaranteed? → No; readiness + verification.
- Will it break? → Conflict control + rollback.
- Ranking? → Technical base; not a promise.
- How fast? → By scope.
- Live tests? → Post-deploy.

Suggested visual: VA-07

Implementation note: Double-CSP warning in a technical box; 'A+ not guaranteed' in FAQ.

AI / GEO / AEO Readiness

Meta title: AI / GEO / AEO Readiness

Meta description: Make your content readable for answer engines: entity, schema, llms.txt, FAQ.

Hero: Readiness for the answer era — Not a citation guarantee; readability engineering.

CTA: See the Readiness Signals

Sections:

- Identity — Entity clarity + sameAs.
- Structure — Schema + semantics.
- Directives — llms.txt + crawler rules.
- Content — Answer-ready FAQ.
- Boundary — No recommendation guarantee.

FAQ:

- Guaranteed outcome? → No; signals are prepared.
- Measurement? → Via signal audit.
- Necessary? → Part of visibility strategy.
- Multilingual? → With hreflang discipline.
- Timeline? → By scope.

Suggested visual: VA-06

Implementation note: Signals as icon+text; the no-citation line visible.

Deployment and Verification Support

Meta title: Deployment and Verification Support

Meta description: GoDaddy/Cloudflare upload, cache management, live header and preview verification.

Hero: Not just upload — evidenced upload — The 7-step Deploy-Verify flow with an expert.

CTA: See the Flow

Sections:

- Upload — Files + .htaccess placement.
- Cache — Hosting/CDN purge procedure.
- Live fetch — Header/redirect confirmation.
- Preview — Debugger refresh.
- Report — Baseline-diff evidence.

FAQ:

- Who uploads? → Expert/owner together.
- DNS? → Owner via recipe.
- What's proven? → The live effect.
- On issues? → Rollback ready.
- Cadence? → As needed.

Suggested visual: VA-30

Implementation note: A 7-step timeline; rollback assurance in the close.

SSS

Meta title: FAQ — Service Scope and Boundaries

Meta description: Honest answers on process, scope, boundaries and verification.

Hero: Answers before you ask — A service FAQ in claim-safe language.

CTA: Full FAQ

Sections:

- Scope — Size/languages/modes/monitoring.
- Process — Review→SafeFix→Verify→Monitor.
- Boundaries — No guarantees; evidence instead.
- Roles — Engine/expert/owner.
- Start — With Review.

FAQ:

- Price? → Discussed by scope.
- Timeline? → By size.
- Guarantees? → Readiness + evidence.
- Who executes? → Clear via the role table.
- First step? → Review.

Suggested visual: VA-25

Implementation note: Category accordions; answers within 2-3 sentences.

22. Use Cases

Figure VA-40 — The customer journey: five stops from review to verified deployment.

Corporate site pre-redesign

Before the redesign budget is approved, Review produces the evidence snapshot of the current asset: what is worth preserving and what is debt. After the redesign, the same baseline is the measurement ground for 'did it really improve'.

Typical evidence output: Baseline report + keep-list

First step: Review — with the current ZIP

Live-site trust readiness

For a live site, SafeFix + Deploy-Verify raise readiness from security headers to social preview without risk; every step moves under rollback protection.

Typical evidence output: Fixed ZIP + rollback + diff report

First step: Review + SafeFix scope approval

SEO / GEO / AEO readiness

Before content investment, the technical base and AI-readability signals are established; investment lands on crawlable, answerable ground.

Typical evidence output: Technical-base score + AI-signal list

First step: Review — before the content plan

Multilingual site audit

Audits hreflang, identity consistency and per-language meta discipline in one report; silent drifts between language copies become visible.

Typical evidence output: Language-consistency table + hreflang findings

First step: Review — with all language copies

Agency delivery audit

The delivery ZIP goes through Review: promised quality is confirmed with evidence, gaps are classified; acceptance rests on a document.

Typical evidence output: Acceptance document: findings + class + decision

First step: Review — on the delivery ZIP

Investor / corporate trust document

The evidence package (scores, changed files, verification plan) attaches to the due-diligence file; 'the web asset is governed' becomes a document.

Typical evidence output: Evidence-package annex (scores/files/plan)

First step: Packaging the existing baseline

Domain portfolio

Many domains are compared in the same 10-domain frame; investment and maintenance priority is ordered by data.

Typical evidence output: Portfolio comparison table

First step: A short Review per domain

Campaign landing page

Before heavy traffic, speed readiness, preview integrity and tracking hygiene verify in one run; campaign-day surprises drop.

Typical evidence output: Speed+preview+tracking verification report

First step: One run on the campaign page

Media / finance / technology site

In high-trust sectors, header policies, schema identity and privacy visibility are audited on a regular rhythm; reputation risk shows early.

Typical evidence output: A series of periodic diff reports

First step: Establishing the Monitor rhythm

23. In-Guide FAQ — 20 Critical Questions

Q: What can it see without touching the live site?

A: All in-file signals in ZIP/local input: meta, schema, header readiness, accessibility markers, static security patterns.

Q: ZIP vs live-URL analysis?

A: ZIP shows file truth; live URL shows response truth; headers and redirects finalize only live.

Q: What if the site breaks after SafeFix?

A: The rollback manifest reverts in one step; the backup dir ships in the package.

Q: How is the rollback manifest used?

A: Copy files back from the backup dir; the changed/created lists guide you.

Q: Which changes need human approval?

A: Meaning-changing content, brand/voice decisions, legal text and the publish decision.

Q: Why isn't the 2,033 catalog automated checks?

A: The catalog is a signal map; a check is code the engine runs. The two are deliberately separate.

Q: Why is 319 the more accurate number?

A: Counted from code, consistent with CLI/gate/registry, and free of inflation.

Q: What ROADMAP=0 is not?

A: Not 'everything automated'; every item attached to its correct artifact class.

Q: Is a runtime bridge a real measurement?

A: The bridge routes to a real tool; the tool measures, the engine fabricates nothing.

Q: Who executes external action recipes?

A: The owner or authorized ops; the engine writes instructions, never claims 'done'.

Q: Does the engine change Cloudflare settings?

A: No; it gives a step-by-step recipe.

Q: Does the engine upload to GoDaddy?

A: No; it provides upload instructions and the verification flow.

Q: Is the SecurityHeaders grade guaranteed?

A: No; readiness is produced, the grade is given live by the tool.

Q: Is the SSL Labs grade guaranteed?

A: No; it depends on server config and is measured live.

Q: Is the PageSpeed score guaranteed?

A: No; it depends on lab/field conditions.

Q: Does the engine produce CWV field data?

A: No; it comes from CrUX/PageSpeed.

Q: Any AI citation guarantee?

A: None; readability readiness instead.

Q: What is llms.txt for?

A: Access/summary directives for AI crawlers; supports identity clarity.

Q: What if JSON-LD is wrong?

A: Rich-result eligibility can break; hence type-page fit is audited.

Q: When is Build mode needed?

A: When new surfaces/pages are needed while preserving existing value.

The full FAQ (85+ questions) is in the faq/ folder.

24. Appendices

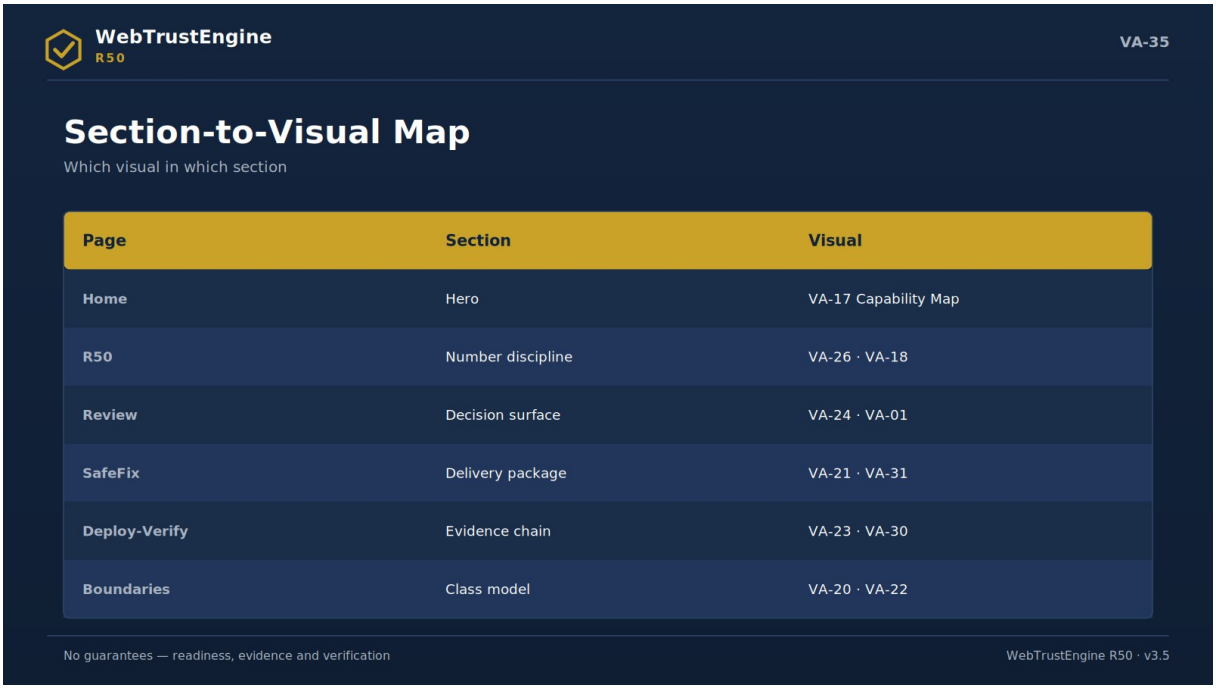

Figure VA-35 — The section-to-visual mapping: a summary map of web placement.

Figure VA-37 — The content library: six scale items from the guide to the banner system.

Number Origin Ledger

Distinct working: 80 core + 239 granular = 319 implemented detectable checks; 68 security patterns; 26 SafeFix generators; 21 runtime bridges; 24 external-action recipes. Reference catalog 2,033; distribution ENTEGRE 1,258 · RUNTIME 197 · EXTERNAL 308 · CONDITIONAL 214 · MANUAL 55 · N/A 1 · ROADMAP 0. CLI/gate/registry agree (80 checks / 26 fixers).

Tool Ecosystem

SecurityHeaders · Mozilla Observatory · SSL Labs · PageSpeed · Lighthouse · CrUX · WebPageTest · Screaming Frog · Search Console · Bing Webmaster · axe-core · Rich Results · Schema validator — OWASP · WCAG 2.2 · Schema.org · Core Web Vitals · W3C · llms.txt/AI-GEO.

Category Dictionary

- ENTEGRE — catalog row functionally covered by a working check
- RUNTIME-BRIDGE — requires live tool/measurement
- EXTERNAL-RECIPE — host/DNS/account action
- CONDITIONAL-RULE — valid by page/sector type
- MANUAL-VERIFICATION — requires human judgment
- NOT-APPLICABLE — not applicable in this context

Deployment Checklist

- fixed ZIP uploaded to hosting
- .htaccess at root and being read

- hosting cache purged
- CDN cache purged
- live headers compared against the expected set
- HTTP→HTTPS redirect tested
- 404 behaviour tested
- sitemap.xml accessible
- robots.txt accessible
- llms.txt accessible
- OG/Twitter debugger refreshed
- favicon/manifest visible live
- visual check done on key pages
- rollback package archived
- Deploy-Verify report produced

External Validation Checklist

- SecurityHeaders scan run
- Mozilla Observatory run
- SSL Labs assessment obtained
- PageSpeed/Lighthouse lab pass run
- CrUX/field data checked (if any)
- Search Console verification done
- Bing Webmaster verification done
- Rich Results test run
- social preview inspectors checked
- redirect chain traced
- broken external-link scan run
- results bound to baseline

25. Board Questions and Answers

Figure VA-38 — Management decision questions: claim-safe answers to six critical questions.

The ten most frequent board questions, answered in claim-safe language:

Q: What does this investment gain us?

A: It turns an unmeasured cost item (the web) into a measurable asset: scores, changed files, verification evidence. Decisions rest on tables, not feelings.

Q: Is our risk decreasing?

A: The static security review surfaces silent risks (secret leaks, exposed files, risky patterns); header recipes prepare to narrow the browser-side attack surface. The proof of reduction is before/after plus independent verification.

Q: Why not just buy a pentest?

A: You can — this product does not replace one and never claims to. Static review is continuous low-cost hygiene; a pentest is authorized periodic deep testing. They complement each other.

Q: Will we rise on Google?

A: No ranking promise is given. What is given is crawlability, consistency and readability readiness. Your content investment lands on solid technical ground.

Q: Will AI recommend us?

A: It cannot be guaranteed; no honest vendor can. What is done is building the signals that let AI systems read you correctly.

Q: Who is responsible if the site breaks?

A: Every change ships with a rollback manifest; one-step reversal exists. Publishing is always a human decision — the responsibility chain is clear.

Q: Does it conflict with our agency?

A: No; it complements as the evidence layer. The agency's delivery goes through Review and work is accepted with evidence — the relationship sheds ambiguity.

Q: Ongoing cost or one-off?

A: Governance is cyclical: Review→SafeFix→Verify→Monitor. Scope components (size, languages, modes, monitoring cadence) set the rhythm; figures sit outside this document.

Q: How is it different from competitors?

A: Number discipline: it never sells 2,033 as 'checks', counts 319 from code, and never translates ROADMAP=0 as 'everything automated'. The honest boundary is a feature for enterprise buyers.

Q: What do we see in the first 90 days?

A: The baseline report, the first SafeFix package, a live verification pass and the first Monitor comparison: four concrete evidence files.

26. Implementation Roadmap — Four Phases

Figure VA-11 — The managed service workflow: five steps from initial review to verification and reporting.

Phase 1 · Baseline (week 1)

Input is collected (ZIP/folder/authorized URL), Review runs, the 10-domain score and finding classification emerge. Output: the decision surface + suggested SafeFix scope + external-action list. No file changes in this phase.

Phase 2 · SafeFix (week 2)

Approved-scope low-risk fixes are applied on a working copy. Output: fixed ZIP + changed-file list + rollback manifest + before/after internal score. Approval-needing items wait on a separate list.

Phase 3 · Deploy-Verify (week 3)

The package is uploaded, caches purged, live headers and previews verified, the independent-tool pass runs. Output: the baseline-diff report + tool-bridge results. 'Produced' becomes 'working' here.

Phase 4 · Monitor (ongoing rhythm)

At the agreed cadence (monthly/quarterly) the live state is compared with baseline; drifts, new risks and external-action status are reported. The loop moves the web asset from 'project' to 'governed asset'.

Note The timeline is representative; it scales with site size and scope components.

27. Role and Responsibility Model

'Who does what' is governance's most practical question. The table below clarifies the boundary between engine, expert and site owner.

Work item	Engine	Expert	Owner
Static detection + scoring	Engine	—	—
SafeFix production + rollback	Engine	scope approval	—
Manual-verification items	flags	evaluates	approves
Hosting upload	writes instructions	assists	executes/authorizes
DNS/CDN/email records	writes recipes	guides	applies in panel
Live tool passes	bridges	runs/interprets	grants access
Publish decision	—	advises	decides

This model removes both the 'the engine does everything' illusion and 'who was responsible' ambiguity; every row ports verbatim into contract language.

28. R50 Glossary

- Baseline — The starting snapshot Review produces; the reference for all later comparisons.
- SafeFix — The mode producing low-risk, reversible fix packages.
- Rollback manifest — The reversal document holding the backup dir plus changed/created file lists.
- Changed files — The full list of files SafeFix touched; the core of evidence.
- Evidence package — The delivery set of scores, file lists, recipes, QA and checksums.
- Runtime bridge — The bridge binding a non-static check to a real tool.
- External action recipe — The recipe turning panel/DNS/account steps into actionable instructions.
- Conditional rule — A rule valid only when the page/sector applies (e.g. Product schema).
- Manual verification — The item class needing human judgment; never auto-PASSed.

- Implemented detectable check — A detection check the engine actually runs, counted from code.
- Reference catalog — The 2,033-item signal map; not a check count.
- ENTEGRE — The status of a catalog row functionally covered by a working check.
- ROADMAP — The count of unclassified items; zero in R50.
- Claim-safety — The language discipline that promises no unverifiable outcome.
- Deploy-Verify — The verification loop proving live effect after upload.
- Diff report — The table of differences between baseline and live state.
- Entity clarity — Machine-level clarity of who/what/where.
- llms.txt — The file giving AI crawlers access/summary directives.
- Structured data — The JSON-LD layer making content machine-readable.
- Canonical — The tag declaring the page's primary URL; prevents duplicate signals.
- hreflang — The tag set mapping language/region versions.
- HSTS — The header instructing browsers HTTPS-only.
- CSP — The policy limiting which resources a page may run.
- SAST — Static security analysis over source without touching the system.
- DAST — Active testing on a live system; only with authorization+scope.
- OWASP mapping — Mapping findings to the shared security language.
- Core Web Vitals — LCP/INP/CLS field-experience metrics; tools measure them.
- Lab vs field — The split between controlled tests and real-user data.
- Cache purge — Clearing hosting/CDN caches to enable live verification.
- Preview debugger — The platform tool refreshing social-preview caches.

29. Common Mistakes and the Correct Approach

The fourteen mistakes below are the most frequent and costly in the field; each comes with its correct approach.

Mistake: Selling the catalog count as 'checks'

Correct approach: 2,033 is a reference map. Correct: 319 working checks; the map is explained separately.

Mistake: Promising a live A+

Correct approach: The tool grades live. Correct: produce readiness, verify independently post-deploy.

Mistake: Treating file production as done

Correct approach: Cache and server layers intervene. Correct: the Deploy-Verify pass is mandatory.

Mistake: 'Solving' CSP with inline onload

Correct approach: Strict CSP breaks the page. Correct: defer external scripts; avoid inline tricks.

Mistake: Creating a double CSP

Correct approach: Server+file policies intersect and narrow. Correct: detect the existing policy; keep one source.

Mistake: Stamping Product/Event schema everywhere

Correct approach: Wrong types break rich results. Correct: conditional rule — schema only when content exists.

Mistake: Leaving empty alt on non-decorative images

Correct approach: Information loss and accessibility debt. Correct: meaningful alt; empty only when decorative by intent.

Mistake: Letting robots contradict the sitemap

Correct approach: Blocked URLs remain in the sitemap. Correct: generate both from one source in sync.

Mistake: Mistaking static review for a pentest

Correct approach: Scope and law differ. Correct: written SAST ↔ DAST split; active tests with authorization.

Mistake: Reporting lab scores as field CWV

Correct approach: They are different truths. Correct: lab readiness and field data in separate columns.

Mistake: Counting DNS work as done

Correct approach: Records don't change until the recipe is applied. Correct: external action + verification step.

Mistake: Auto-PASSing manual items

Correct approach: The honesty layer is breached. Correct: flag, justify, route to a reviewer.

Mistake: Forgetting preview caches

Correct approach: Stale cards circulate for days. Correct: debugger refresh is part of the procedure.

Mistake: Quoting numbers without provenance

Correct approach: Every number gets challenged. Correct: the number ledger + registry row evidence.

30. Scope Components Guide (No Pricing)

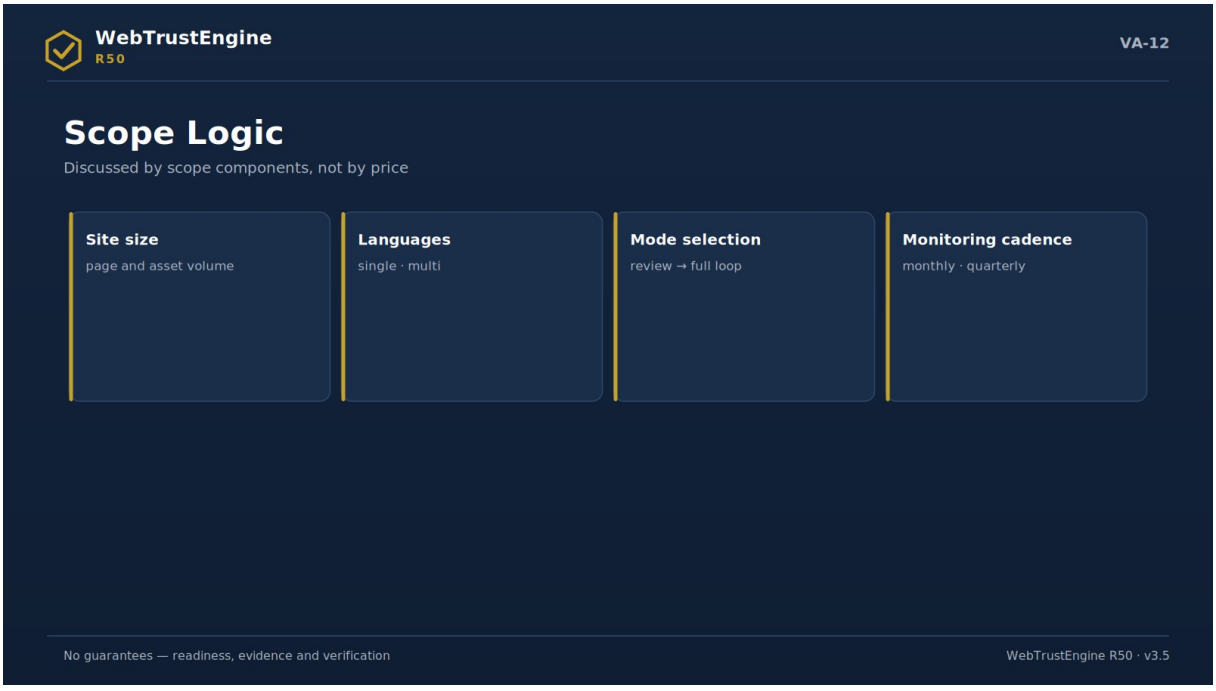

Figure VA-12 — Scope logic: size, languages, modes and monitoring cadence.

Scope is discussed via four components; none maps to a figure, since this document is public.

Site size

Page count and asset volume; scales Review time, finding density and SafeFix scope. A small corporate site and a multi-section portal do not share a rhythm.

Language count

Every language copy brings its own meta/schema/hreflang discipline; consistency auditing grows with languages.

Mode combination

Review only; Review+SafeFix; the full loop (including Build and Monitor) — three depths of engagement.

Monitoring cadence

Monthly, quarterly or release-based Monitor rhythm; chosen by your rate of change.

31. How to Read the Reports

Score table

10 rows, internal readiness out of 100. Reading order: the three lowest domains → their entries in the findings list → which class (file/live/external/manual).

Findings list

Domain · severity · file · class columns. 'High severity + ENTEGRE' rows are first SafeFix candidates; 'MANUAL' rows go to the approval queue.

Rollback manifest

Verify the backup_dir path and the two lists (changed/created); archive it. A reversal drill is five minutes of cheap insurance.

Deploy-Verify diff report

Compare expected ↔ live columns; an 'expected present, live missing' row is a cache or server-processing issue — check purge first.

32. Release and Change Discipline

R-releases (R49, R50…) mark the engine's capability moments; each ships with patch notes, a test report and a count report. The frozen core is never altered between releases; new capability lands in a separate layer. Marketing documents carry the same discipline: every version has a CHANGELOG, QA report and checksums — this guide, as v3.7, is part of that chain.

The public value of this discipline: any number, claim or screenshot can be verified against the evidence file of the release it belongs to; 'which version was that' ambiguity disappears.

33. Evidence Package File Dictionary

The files you will meet in the delivery ZIP, each with its one-line duty:

- RUN_SUMMARY.json — The run's summary record: input, mode, page count, before/after score.
- SKOR_RAPORU_*.md — The 10-domain table + fix types + external-work list.
- FIX_MANIFEST.json — Type-and-count breakdown of applied changes.
- ROLLBACK_MANIFEST.json — backup_dir + changed + created file lists.
- DEGISEN_DOSYALAR_*.txt — The human-readable changed-file list.
- EXTERNAL_ACTION_RECIPES.md — Panel/DNS/account recipes under 24 headings.
- RUNTIME_BRIDGE_KONTROLLERI.md — The tool/metric/how map of the 21 live checks.
- MANUAL_VERIFICATION_RULES.md — Items awaiting human approval, with rationale.
- GODADDY_YUKLEME_TALIMATI.md — Upload + cache + verification steps.
- MANIFEST_*.csv — The file/size/type inventory of the package.
- CHECKSUMS_*.sha256 — Integrity digest of every file; produced after packaging.
- QA/PATCH/COUNT reports — The release's quality, change and count evidence.

34. Working Protocol with Independent Tools

Figure VA-04 — External measurement ecosystem: independent tools produce live scores; the engine binds results via bridges.

Figure VA-36 — The tool-to-domain matrix: the domain and timing map of independent validation.

Run-and-read notes for the six main tools used in the Deploy-Verify pass:

SecurityHeaders

Enter the domain, read per-header; gaps map to the .htaccess recipe. If the grade is low, check cache/processing first.

Mozilla Observatory

Compare the post-scan suggestions with our header recipe; overlapping advice merges into one policy.

SSL Labs

Full analysis takes 1-2 minutes; read chain, protocol and OCSP rows. Server-side findings go to external recipes.

PageSpeed / Lighthouse

Write lab metrics into the readiness column; field (CrUX) data is a separate column. Never report the two as one score.

Rich Results Test

Test schema-bearing sample URLs; on errors re-audit type-page fit and required fields.

Search Console

After verification submit the sitemap; bind coverage and CWV reports to the Monitor rhythm.

35. Sector Application Notes

The same frame needs different emphasis by sector; the notes below say where to look on the first run.

Finance / investment

Security headers and privacy visibility lead; claim language (returns/success) passes the claim-safety filter; legal texts go to the manual queue.

Media / news

Structured data (Article/Person) and author-date signals are critical; social-preview integrity binds to Deploy-Verify at each publish.

Technology / SaaS

Heading hierarchy and code-sample accessibility on docs pages; AI readability (llms.txt) should be set early.

Healthcare

Content accuracy centers the manual layer; schema is built to avoid medical claims; privacy signals are tightly audited.

Education

Multilingual consistency and the static accessibility set lead; event/course schemas are added via conditional rules.

E-commerce (info layer)

Product schema only on real product pages; performance readiness and preview cards bind to the campaign rhythm.

Public / NGO

Accessibility and transparency texts take priority; document/PDF link hygiene is specially flagged in Review.

Agency / studio portfolio

Its own site is showcase evidence: a sample evidence package is published; before/after table language replaces claim numbers.

36. Evidence-First Communication Templates

Five short copy-ready templates; all pass the claim-safety filter.

Executive email

Subject: Web asset — baseline report ready. Body: The 10-domain score, top three findings and suggested SafeFix scope are attached; no files were changed, and I prepared two options for the decision meeting.

Proposal paragraph

The engagement covers the Review baseline, a SafeFix package within approved scope, a Deploy-Verify pass and the first Monitor comparison. Deliverables ship as an evidence package; no ranking/traffic guarantee is given.

Acceptance note

The delivery ZIP passed Review; findings were classified and acceptance criteria met. Remaining external-action items are assigned in the attached recipe.

Status update

This month: 2 SafeFix runs, 41 files changed (rollback archived), live verification clean; of the external actions, DMARC was applied in the panel — verification attached.

Press line

WebTrustEngine R50 is a web governance engine that evaluates web assets across 10 domains with evidence and verifies change after deployment.

37. The 90-Day Measurement Plan

What is measured when, with which evidence, in the first quarter:

Period	Measured	Evidence
Day 0-15	Baseline score + finding classification	Review report
Day 15-30	Internal score delta after SafeFix	before/after + rollback
Day 30-45	Live header/preview confirmation	Deploy-Verify diff report
Day 45-60	Independent tool-pass results	SecurityHeaders/SSL Labs/PageSpeed bridges
Day 60-75	External-action application status	recipe + panel verification
Day 75-90	First Monitor comparison	periodic diff + drift list

After ninety days four evidence files exist and the loop is now the organisation's rhythm; what follows is a decision of scale and depth.

38. Vendor Evaluation Questions

Procurement or internal audit can ask any web vendor these ten questions; what a good answer looks like is given alongside. WebTrustEngine answers them with its own evidence package.

Question: How do you prove your check count?

Good answer: A good answer is code-derived counting plus registry consistency; not a slide number.

Question: Do you provide the list of changes?

Good answer: A good answer is a file-level changed list + rollback; not 'improvements were made'.

Question: What is your reversal plan if the site breaks?

Good answer: A good answer is a backup dir + a rehearsed one-step reversal.

Question: Who produces the live scores?

Good answer: A good answer names independent tools and the bridging logic; not self-scoring.

Question: What can you not automate?

Good answer: A good answer is a clear manual-item list; 'everything' is a red flag.

Question: What is the legal scope of your security testing?

Good answer: A good answer is the SAST/DAST split plus the authorization requirement.

Question: How do you handle DNS/CDN work?

Good answer: A good answer is a recipe with steps+verification; not 'we handled it'.

Question: Do you guarantee ranking/traffic?

Good answer: The good answer is no; readiness and evidence is the correct language.

Question: How do I verify delivery integrity?

Good answer: A good answer is a manifest + checksums; not a lone zipped folder.

Question: What will we measure in ninety days?

Good answer: A good answer is a dated measurement plan mapped to evidence files.

39. Quick Start: Preparing the First Review

Eight preparation items to get the most from the first run:

- The full site ZIP: From the root, with all assets; a partial folder yields missing-signal noise.
- The domain list: Primary plus subdomains if any; needed for identity consistency.
- The language inventory: Which languages are live; the hreflang audit builds on it.
- The access note: If live-URL mode is wanted, the ownership/authorization letter is prepared.
- Existing policy info: Any server CSP/redirects are declared; double-policy risk is avoided.
- The priority sentence: 'Our most critical domain this quarter is…' — score reading orders accordingly.
- Decision owners: Who approves the SafeFix scope; known from day one.
- The archive location: Where evidence packages live, and the versioning layout.

With these eight ready, the first Review can run the same day; the decision surface is on the table within the first week.