

The measured standard

Maxwell Connolly · August 2026

SUMMARY

This is the fourth of four papers. The third described the delivery pipeline and the gates every change passes through; this one goes inside the security standard those gates enforce. Its subject is the measurement itself: the four external references that define the bar, the automated scorecard the estate scores against and why that score is adjusted, the watchers that read every repository on a schedule to confirm each control is still enforcing, the reports they emit and the discipline that they are read, the rolling re-measurement that keeps the figures current, and the safety nets that make a green result mean what it says. The organising principle is that the posture is measured rather than asserted. Every figure below was read from a command-line run or a configuration file on a dated measurement in August 2026.

1 What this is

Kemily runs a real security standard, measured rather than recalled. Four external references define the bar; a machine checks each control on every change; and a second machine reads every repository on a schedule to confirm the first still enforces.

This paper describes that measurement: the automated scorecard the estate scores against and why that score is adjusted, the checks that run on every change, the watchers that audit enforcement on a clock, the reports they emit and who reads them, the rolling re-measurement that keeps the figures current, and the safety nets.

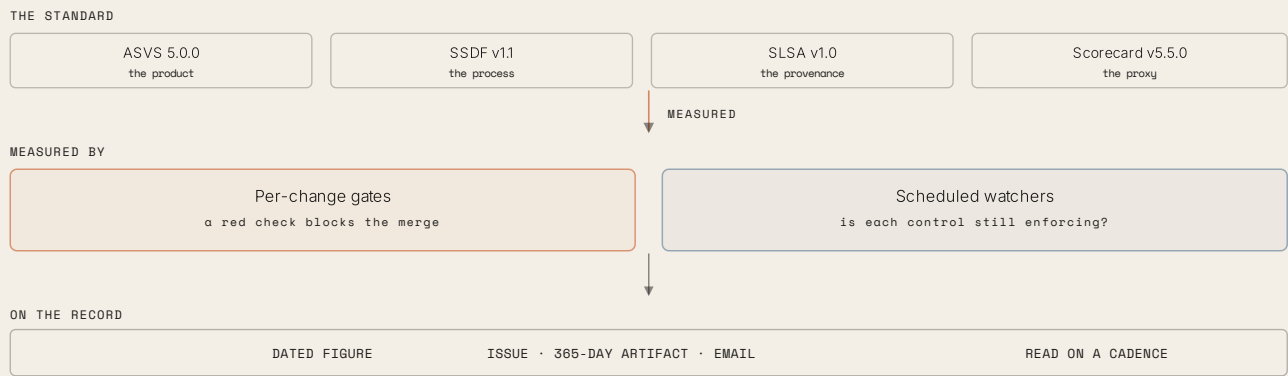


Figure 1 — The regime, in three bands. Four external standards define the bar. It is measured two ways at once: gates that block a change on every pull request, and watchers that read the estate on a schedule to confirm the gates still enforce. Every result lands on the record as a dated figure, emitted through three channels and read on a cadence.

2 The four standards, measured not asserted

The standard is not one document but four external references, each answering a different question, each pinned to a named version so a requirement number cannot drift under it.

STANDARD	VERSION	WHAT IT ANSWERS
OWASP ASVS	5.0.0	Is the product sound — 345 requirements across 17 chapters.
NIST SSDF	SP 800-218 v1.1	Is the development process sound.
SLSA	v1.0	Can we prove what we built came from the source we claim.
OpenSSF Scorecard	v5.5.0	An automated proxy, useful for direction rather than for judgment.

Two disciplines travel with every figure. A Scorecard number is quoted only from a command-line run against our own private repositories, never from the hosted service that scores public repositories — a result quoted without a CLI run behind it is a result for someone else’s repository. And a statement of which checks a branch requires names the mechanism it was read from and the date, because a repository host enforces branch rules through two independent systems and the effective requirement is their union, which no single query returns. A table of required checks without its mechanism cannot be re-derived, and reading one system yields a confident wrong answer.

The version numbers carry the same weight. ASVS 5.0.0 renumbered every requirement from its previous edition, so an identifier copied from a pre-2025 source is invalid until re-checked; and the SSDF has a later draft that is not yet final, which the estate does not cite as normative. Naming the version is part of measuring rather than asserting.

3 The adjusted Scorecard, and why we adjust it

The OpenSSF Scorecard scores a repository from 0 to 10 across a fixed set of automated checks. The estate holds itself to a bar of **8.0** — on a Kemily-adjusted score, with the raw score published beside it so nothing is hidden.

The adjustment removes three checks, and only three, each because it measures something a single-operator, private, proprietary organization structurally cannot do. Left in, they report on the shape of the organization rather than on the security of its code.

EXCLUDED CHECK	WHAT IT ACTUALLY REQUIRES
Code-Review	An approver who is not the author. There is one seat, so nothing can be approved by a second party.
CII-Best-Practices	The OpenSSF badge, which is issued only to a publicly reachable project. The repositories are private.
Contributors	Three or more contributing organizations. There is one.

A check is excluded only when it is unreachable by construction; a check that can be moved is moved instead. Fuzzing was excluded once, and the exclusion was withdrawn a day later after reading the vendor documentation: the tool supports private repositories, so it was a control Kemily could adopt rather than one it could not reach. It was adopted, and the check now scores. Excluding a check the estate can actually satisfy would be gaming the score.

The bar sits on the adjusted score rather than the raw score for a measured reason, not a convenient one. The **raw ceiling — every reachable check at its maximum — is 7.70**, which is below 8.0, so a raw bar of 8.0 would be unreachable by construction. The adjusted ceiling is **8.81**, and the 8.0 bar sits about 0.8 below it, with headroom. Every aggregate is recomputed from the per-check scores using the vendor's own risk weights (Critical 10, High 7.5, Medium 5, Low 2.5), and each recomputation reproduces the reported figure exactly, so the adjusted arithmetic applies the real formula rather than approximating it.

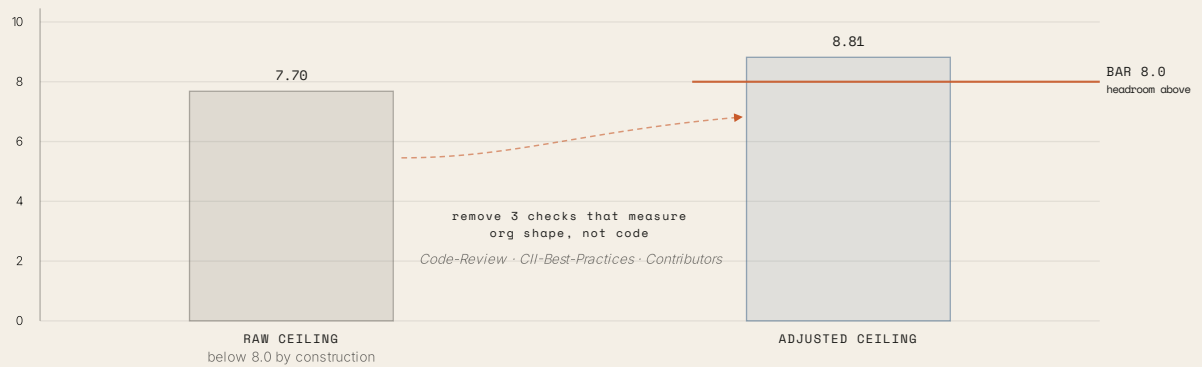


Figure 2 — Why the bar sits on the adjusted score. The raw ceiling reaches 7.70 — below 8.0, so a raw 8.0 bar is unreachable by construction. Removing the three checks that measure organizational shape rather than code lifts the reachable ceiling to 8.81, and the 8.0 bar sits below that with headroom. The raw score is published beside the adjusted one at every measurement.

The raw score is published next to the adjusted one at every measurement, and where the two diverge the divergence is itself on the record. The estate’s static analyzer runs as a required, unbypassable gate on every change, but the Scorecard check that would credit static analysis does not recognize that particular tool and reads zero — so the raw score understates the posture, and publishing it is what lets a reader see that rather than take it on trust.

One of the largest near-term movements in the score arrives on a calendar rather than through work: a check that reads zero on a repository younger than 90 days is worth close to a full point, and it resolves as the repositories age. It is recorded here before it happens, so that when the number moves the cause is already on the record rather than claimed as remediation.

4 The mechanisms — what runs on every change

The standard’s enforcement surface is the pull request. A change reaches a protected branch only through one, and the request merges only when a defined set of checks pass. The checks are required in branch protection, so a red check blocks the merge, and a workflow that runs without blocking is treated as absent.

CHECK	WHAT IT BLOCKS
Static analysis (Semgrep)	Dangerous code patterns fail the build on a finding, and the scan cannot be silently skipped.
Secret scanning (gitleaks)	A committed credential fails the build; it cannot reach the default branch.
Dependency scanning (osv-scanner)	A known-vulnerable dependency cannot merge; a release-age cooldown holds back freshly published versions.
Typecheck and build	Code that does not type or does not build is refused.
Coverage floors	A drop below the configured floor fails the run instead of printing a report.
Cross-tenant isolation	On the multi-tenant products, a change that lets one tenant reach another's data fails against a policy-capable database.
Browser end-to-end	The behavioral flows are exercised in a real browser in CI.
Continuous fuzzing	Parsing and boundary code is fuzzed on every change — Jazzer.js for TypeScript, Atheris for Python; a crashing input fails the job.

Two properties keep the set honest. A branch's true required set is the union of two independent enforcement systems, and the estate reads that union rather than one side of it. And each check's context is pinned to the application that produces it, so a token that can post a status cannot forge a passing result the scanner never emitted.

5 The watchers — the machines that audit the machines

Enforcing a control and proving it still enforces are two different jobs. A required check that quietly stops running still reports green, which is the failure a passing dashboard hides. So a second tier reads the estate on a schedule and asks, per control, whether it is still enforcing.

- **The conformance sweep runs daily.** It reads every repository against seventeen controls and states each as compliant, drift (present but not enforcing), gap (absent and relevant), or not-applicable (declared, with a machine-checkable premise). It also reads the project template itself, so a control silently dropped from what new repositories inherit is caught by machine rather than by a later audit.
- **The rolling Scorecard runs weekly.** It re-runs the command-line tool, recomputes the adjusted score from the per-check scores, and signals on a drop below the bar or a check that regressed against the previous run.
- **The document-control job holds the canon to its own clock.** Each security document carries review and retention triggers in machine-readable front-matter,

and the job re-checks them on schedule, so the documents cannot silently age out of currency either.

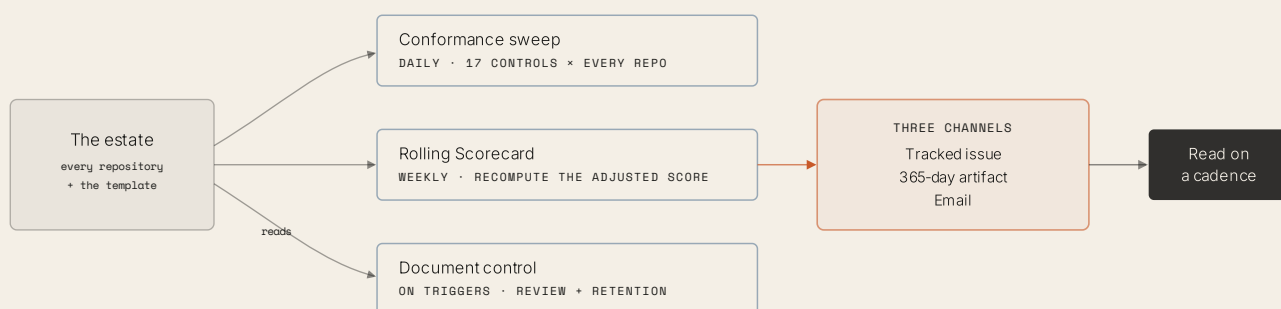


Figure 3 — The watchers. Three scheduled machines read the estate, each asking whether a control is still enforcing rather than merely present. Each emits the same three channels — a tracked issue, a retained artifact, and an email — and the result is read on a cadence, because a control nobody reads is not a control.

6 The reports, and the discipline that they are read

Each watcher emits the same three channels, so a result cannot exist only in a log that no one opens: a tracked issue that accumulates the rolling triage, an artifact retained for 365 days, and an email. The subject line carries the verdict, so the outcome is known before the report is opened — a clean sweep, a count of actionable findings, or, treated as the hardest failure of the three, an incomplete read that could not be trusted whole.

The reports are read on a cadence, not when someone remembers: the daily orientation reads the latest conformance result from a fixed path, and a standing calendar review sits against each watcher. A finding is **carded, not noted** — one left off the board is one the next sweep raises again. This is enforced because it once failed: a monthly audit ran for three months, opened an issue and committed an artifact that nobody read, while three real findings sat open under a report that claimed a clean sweep of a fraction of the estate.

7 Rolling management — recheck and adjustment

Nothing in the regime is measured once. The scorecard re-runs weekly and the sweep daily, so the published figures stay current rather than a snapshot that ages out of date.

The bar itself is re-derived rather than fixed. Stated conditions reopen it, each written down as a trigger so it moves when the world does and not when someone happens to notice: a second operator joining, which makes an excluded check reachable and dissolves the exceptions that rest on there being one seat; a Scorecard-recognized static analyzer being

adopted, which recovers the points the current one is not credited for; or a repository passing 90 days, which moves a check on the calendar. The conditions are the schedule; the recompute is mechanical.

The same holds for the documents and the exceptions. Every canon document carries its own expiry triggers, and an exception is valid only while its premise holds — including the estate's own exceptions. Each declared exception carries the condition under which it retires, and the sweep re-checks that its premise is still true, so an exception with a reason that has quietly expired is caught as drift rather than surviving as an oversight.

8 The safety nets

The regime is built around the specific ways a measurement can mislead, and each has a net that turns the failure into a visible signal.

- **Fail loud on a partial read.** An incomplete sweep exits with a failure and the verdict “incomplete” rather than reporting green over the repositories it could not reach, because a partial read that reads as clean hides everything beneath it. The rolling Scorecard's first live run did exactly this, refusing to report a scan it could not prove whole.
- **Prove the guard can fail.** A control is not accepted until it has been observed failing: the build is run with the guard broken to confirm the check goes red, and the red observation is produced by the reviewer rather than the author. Continuous fuzzing earned this standing on adoption, finding a genuine cross-tenant path traversal in one product and an input-handling crash in the pipeline's own tooling on its first runs.
- **Watch the template, not only the repositories.** A gate dropped from what new repositories inherit is caught by a template-parity control that reads the template each day, so the standard cannot stop propagating forward without a machine noticing.
- **Ship every gate with a tested override.** A control that surfaces a finding also ships a documented path to remediate it, so the standard never blocks its own fix. The dependency-freshness cooldown, for one, carries an escape hatch for an urgent security update, so the rule that holds back fresh releases cannot hold back the release that fixes a live vulnerability.
- **A null result is not evidence.** A check that finds nothing may have had nothing to read. Where a claim rests on absence, the regime states what would have produced a positive and confirms the check could produce it, so a scan that is green because it examined an empty target is not mistaken for a scan that is green because the target is clean.

9 Born with the standard

The standard propagates forward through the project template a new repository is created from. The template carries the gate workflows, the fuzzing scaffold, and the dependency and secret configuration, so a repository scaffolded from it arrives with the scanners running on its first commit and the dependency watcher already active.

This was proven, not assumed. A throwaway repository created from the template in August 2026 was born carrying seven security workflows and the fuzz scaffold, with the scanners green on the first push and the dependency watcher already opening update requests. A single wiring step — requiring the checks in branch protection — is deliberately left for creation time, and the conformance sweep names exactly that step rather than passing over it. The standard is uniform across products because it is inherited from one template, verified outward by the sweep that reads every repository against it. It propagates forward and is checked backward, and neither direction depends on memory.

10 Observation

The regime rests on one property: the posture is measured, and the measurement is itself audited. The controls are mechanical, a second machine reads them on a schedule, the adjusted score is published beside the raw one, and every figure carries the date and the tool it was read from.

An enterprise security review asks whether the controls exist, whether they run, and whether something independent checks them. Each answer here is a command output, which is what lets a single operator hold a product to the bar such a review expects.

RELATED DOCUMENTS

Paper 01 — Three products, one pipeline. The products this standard protects, and the divergence in their requirements that makes a common standard worth measuring.

Paper 02 — Helm: an experiment in AI continuity. The coordinator that runs the estate and reads these reports at every orientation.

Paper 03 — The engineering pipeline. The delivery pipeline whose gating-and-testing stage this paper measures from the inside.