

ActionableScore (AS)

An evidence-driven, LLM-friendly triage metric for Linux kernel vulnerabilities

Technical overview | Experimental methodology | October 2026 | kernelcve.org

1. Why ActionableScore exists

CVSS is an essential standardized severity framework, but it is not designed to serve as a complete evidence ledger for Linux kernel patch triage. Two vulnerabilities may receive similar CVSS base scores while differing substantially in the attacker-controlled primitive, the reachable call chain, required setup, or the credibility of exploitation beyond denial of service.

ActionableScore (AS) is an experimental companion metric for practical prioritization: how much independently supported evidence makes a Linux kernel issue actionable, and how urgently does it merit expert investigation? It does not replace CVSS, calculate exploit probability, or dictate a vendor impact classification.

2. Why the emergence of LLMs matters

Modern large language models (LLMs) can inspect commit messages, patches, control flow, subsystem conventions and supporting advisories at scale. Yet a request to “assign CVSS” alone often encourages a leap from a bug label to a severity vector. AS provides a more teachable intermediate task: identify discrete, technically verifiable signals; explain the evidence for each; subtract only proven constraints; and explicitly distinguish a demonstrated primitive from an imagined exploitation chain.

For example, “UAF” is not itself evidence of privilege escalation, “AV:N” is not proof of unauthenticated remote reachability, and “C:H” is not evidence of data disclosure. This evidence-first scoring discipline is useful for LLM-assisted triage, reproducibility, auditing and model calibration. LLM output remains a hypothesis until grounded in patch, call-path or authoritative evidence.

3. The calculation in five steps

- Start with a raw score of zero. Identify the actual bug primitive before judging severity.
- Add independently established positive signals with their published weights.
- Subtract negative signals only when the restriction is positively supported, not merely “typical” or possible.
- Remove duplicate descriptions of the same mechanism; apply any specifically defined evidence-based floor or cap and disclose it.
- Produce a conservative and a technically defensible higher assessment when evidence supports a range. Publish the numeric score alongside its signal trace and separate review decisions.

A generic hand-calculation is: $\text{raw AS} = \Sigma(\text{independent positive signal weights}) - \Sigma(\text{proven negative constraint weights})$. Floors/caps and presentation rules, where applicable, are separate from the raw arithmetic. There is no universal automatic mapping from AS to CVSS or IMPORTANT/MODERATE.

4. Signal catalogue: what can increase AS

Signal	Weight	Evidence threshold / meaning
Direct network trigger	+2	Attacker traffic reaches vulnerable operation without a separate major privileged prerequisite.

Firmware-mediated external influence	+1	Device/firmware mediation is a material prerequisite; not direct network reachability.
Ordinary local unprivileged trigger	+1	Complete vulnerable operation reachable by an ordinary local user.
High-control unprivileged API	+1	Attacker materially controls relevant object layout, content, lifetime or page ownership.
Real UAF / lifetime violation	+2	Post-free or post-retirement use, mutation, or release established by mechanics.
Strong corruption primitive	+2	Controlled write, type confusion, object replacement, directed overwrite or comparable evidence.
Constrained overwrite / indirect corruption	+1	Actual but limited write, invalid structured interpretation or bounded candidate.
Strong exploitation leverage	+2	Controlled reclaim, sensitive overwrite, refcount takeover or demonstrated leverage.
Weak exploitation concern	+1	Specific reuse or attacker-influenced mechanism beyond generic bug terminology.
Reliable host crash / strong DoS	+1	A realistic host-fatal effect; not simply a warning or a CVSS A:H metric.
Distinct lasting availability effect	+1	Independent persistent shutdown/loss, not the same crash counted again.
Concrete persistent resource/reference leak	+1	A real missing release/put that escapes normal cleanup.
Plausible confidentiality / integrity	+1 each	Concrete disclosure or unauthorized protected-state modification path.
Authorization decision bypass	+2	A demonstrated DENY-to-ALLOW security decision failure.
Sensitive FD, file or credential access	+2	Distinct protected-resource consequence established.
Practical exploit chain	+1	Evidence connects the trigger to meaningful security impact beyond the counted crash.
Repeated release / refcount mutation	+1	Same potentially dangling reference undergoes a second lifetime operation.
Shared-page / COW ownership violation	+2	Write to permitted A changes independently protected B via shared backing.

Filesystem/storage path relevance	+1	Independent significant filesystem, mapping, journal or writeback exposure.
Remote direct kernel-fault primitive	+1	Unauthenticated input controls a concrete fault-prone kernel operation; special floor may apply.
Cross-object live-state interference	+1	One active object concretely redirects/erases state owned by another.
Dirty-Pipe-like weaker primitive	+1	Shared file/page corruption concern short of proven full ownership violation.

5. Evidence-backed reducing signals

Constraint	Weight	When it is legitimate
Host-admin-equivalent privilege required	−2	Full trigger positively requires host root/admin-equivalent control.
Reduced/delegated capability required	−1	A proven narrower but material privilege gate.
Rare and difficult-to-reach setup	−1	Both genuinely uncommon and not ordinarily attacker-instantiable.
Legacy/non-default execution mode	−1	Proven requirement for a legacy/fallback mode.
Genuinely rare physical access/hardware	−2	Uncommon physical-only prerequisite demonstrably limits exposure.
Hard trigger conditions	−1	Specific unusual packet/race/setup conditions supported by evidence.

UNKNOWN is not a penalty: uncertain host-admin requirements do not justify −2; a niche subsystem does not automatically justify −1; a namespace name in a reproducer does not prove that all capability checks are namespace-scoped. Conversely, missing proof of a restriction is not proof of unprivileged reachability.

6. Rules that prevent misleading arithmetic

- CVSS values are diagnostic context, not source evidence for AS points. AV:N, PR:L, C:H or A:H cannot independently earn reachability, corruption, disclosure or crash points.
- A concrete UAF is not counted again as generic corruption. A single panic is not both “reliable crash” and “additional availability.” An authorization bypass and the same downstream effect cannot be repeatedly renamed as separate bonuses.
- Large OOB copy size is not synonymous with attacker control over the destination. Ask which bytes and objects are actually controllable, and whether a useful write occurs before a fault.
- Conservative means restrained about unproven consequences, not an assumption of the most restrictive conceivable deployment. The higher assessment still needs a concrete mechanism.
- ManualReview=REQUIRED and AutoDowngradeAllowed=YES may coexist. Human review does not by itself establish Important-grade exploitation or justify blocking an automatic Important-to-Moderate adjustment.

7. Interpreting the score

AS range	Operational reading
0–2	Low-like or ordinary Moderate: bounded correctness, crash or weak security evidence.
3–4	Borderline: investigate the reachable primitive and missing evidence.
5–6	Actionable Moderate: prioritize review; Important candidate if independent evidence warrants it.
7–8	Strong Important candidate, subject to validation.
9+	High-end Important / possible Critical candidate; not an automatic vendor rating.

These bands are heuristics. PrimitiveClass (NONE, CLASS_C, CLASS_B, CLASS_A), ImportantCandidate, ManualReview, RealUAF, HostAdminRequired, NamespaceScopedPrivilege and AutoDowngradeAllowed communicate decisions that a single number cannot. A strong primitive can require review at AS=5; a crash-only case must not be promoted solely by generic speculation.

8. Two CVEs across evolving ActionableScore revisions (September 4–October 18, 2026)

The October engineering discussion began with a practical question: why was CVE-2026-53151 handled as MODERATE_KPANIC while CVE-2026-80844 was classified IMPORTANT, when both had high and quite similar analyst-reviewed CVSS scores in Red Hat OSIMDB? As checked on October 18, 2026, OSIMDB records CVE-2026-53151 at CVSS 3.1 8.1 (AV:N/AC:H/PR:N/UI:N/S:U/C:H/I:H/A:H), and CVE-2026-80844 at 8.3 (AV:N/AC:L/PR:L/UI:N/S:U/C:H/I:L/A:H). These are the analyst-reviewed OSIMDB values, not the automated model’s paranoid CVSS estimates or the operational severity labels. A Moderate impact classification can therefore coexist with CVSS 8.1; the historical MODERATE7 shorthand is not its actual OSIMDB CVSS. The difference in current integrated ActionableScore—4 versus 5—helps expose why similar CVSS magnitudes need not imply the same kernel triage outcome.

CVE-2026-53151 — AF_RXRPC fragmented ACK/SACK parsing

September 4, 2026 baseline: the archived assessment recorded AS=4 (lower=2) and MODERATE REGULAR for CVE-2026-53151. The older trace attributed +2 to AF_RXRPC network reachability, +1 to an indirect invalid-read candidate, +1 to crash concern, and –1 each to special fragmentation and narrow deployment assumptions; tentative C/I hypotheses appeared only in its higher assessment. That ledger is a historical snapshot, not the arithmetic of the October prompt.

Assessment / observation	Recorded value
Sep 4 archived production	AS 4; lower 2; MODERATE REGULAR
Oct 18 integrated production	AS 4; MODERATE / MODERATE_KPANIC; OSIMDB CVSS 8.1
Oct 18 standalone primary	AS 5; lower 3; CLASS_B; manual review
Technical distinction	Remote AF_RXRPC non-linear skb access; physical OOB write and LPE unproven

October 18, 2026 revision: the integrated production daemon reported AS=4 and a MODERATE outcome (with a MODERATE_KPANIC description), despite the analyst-reviewed OSIMDB CVSS of 8.1. A separate fresh primary AS

assessment returned 5 (lower=3), CLASS_B, ImportantCandidate=YES, ManualReview=REQUIRED and AutoDowngradeAllowed=NO; this standalone result must not be substituted for the integrated production value. Its higher assessment considered constrained skb-data mutation and plausible crash consequences, but no demonstrated physical heap OOB write, controlled reclaim or privilege escalation. The retained production AS=4 reflects a borderline, review-worthy issue rather than a proven Important-grade memory-corruption exploit.

CVE-2026-80844 — IPv6 AH6 routing-header memmove

September 4, 2026 baseline: the archived assessment recorded AS=6 (lower=5), CLASS_A, ImportantCandidate=YES, AutoDowngradeAllowed=NO and IMPORTANT/HIGH. Its earlier ledger awarded extra points for the memory-layout invariant and structured header control and included a higher weak-exploitation concern. Those were historical scoring choices and should not be silently reused as independent October bonuses.

Assessment / observation	Recorded value
Sep 4 archived production	AS 6; lower 5; IMPORTANT
Oct 18 primary	AS 5; raw 2 -> safety floor 5; CLASS_A; OSIMDB CVSS 8.3
Downgrade and preservation	FD proposed MODERATE7; V42 veto; Important wins final merge
Final operational outcome	HIGH / IMPORTANT; manual review required

October 18, 2026 revision: the current primary ActionableScore is 5, CLASS_A, ImportantCandidate=YES, ManualReview=REQUIRED and AutoDowngradeAllowed=NO; the analyst-reviewed OSIMDB CVSS is 8.3. Both raw conservative and paranoid AS scores are 2: +2 for attacker-influenced OOB memmove, +1 for realistic kernel crash/DoS concern, and -1 for CAP_NET_RAW or delegated privilege. A disclosed evidence-based safety floor raises both scores to 5 because a concrete kernel OOB write is established. Locally generated raw IPv6 HDRINCL input with hdrlen=2 and segments_left=255 can drive an approximately 4,064-byte backward memmove through AH6, subject to the required XFRM AH path. Direct remote triggering, attacker-directed adjacent-object overwrite, disclosure and LPE remain unproven; the stronger primitive, not an asserted exploit chain, motivates Important-preservation and manual review.

The October daemon trace also illustrates a changing decision policy: an FD downgrade recommendation was blocked by V42_LEGACY_CLASSA_FD_VETO for the primary CLASS_A/AS=5 case, and the final operational result was HIGH (IMPORTANT). An earlier October run had ended MODERATE_KPANIC before that preservation correction. This does not establish privilege escalation; it prevents automatic dismissal of a concrete attacker-influenced kernel OOB write pending expert review. Across slightly more than six weeks, the archived September 4 AS contrast of 4 versus 6 became an October 18 integrated contrast of 4 versus 5, while the central distinction remained: the first CVE has an AF_RXRPC non-linear skb parsing/read concern without a demonstrated strong write primitive, whereas the second has a concrete attacker-influenced OOB memmove. The original discussion also exposed the limits of simplistic CVSS analogies: PR:N on the first issue does not itself make it more exploitable than the second, and local capability requirements, trigger complexity, demonstrated primitives and full call-path prerequisites must be assessed independently. ActionableScore is deliberately revised as new kernel corner cases reveal double-counting, weak evidence or overly aggressive downgrades; the two dated snapshots document that ongoing calibration, not a fixed or infallible severity formula.

9. Kernel-specific examples that motivate the methodology

Dirty Frag / Dirty Pipe-like shared-page ownership

Consider RXRPC, ESP or XFRM in-place decrypt/write into skb page fragments that are externally owned or shared. The key question is not merely “did bytes change?” but whether an authorized write to object A changes protected object B because COW or shared-page ownership was not honored. A proven A→shared backing→B chain earns the ownership/COW signal (+2), rather than only a generic memory-write label. A weaker Dirty-Pipe-like concern may earn +1, but generic “reflink” or “page cache” vocabulary is insufficient. Protocol rarity must not erase an established ownership violation.

Checksum metadata and downstream trust boundaries

A CHECKSUM_PARTIAL skb may contain attacker-influenced checksum placement metadata. A bounded two-byte checksum write into an IPv4/IPv6 header is not automatically an OOB kernel write. The analyst must trace whether modified length, offset, protocol or routing fields are later trusted by fragmentation, NFQUEUE, XFRM, bridge or host networking paths. A credible downstream privileged consumer and guest/namespace boundary can justify mandatory review; a generic skb keyword cannot establish guest-to-host compromise.

RCU lifetime, qdisc replacement and repeated release

A qdisc schedule replaced while an unprotected reader consumes the retired object is different from a cleanup-only stale pointer that merely faults. Ask whether an attacker can repeatedly replace active structured state, influence the stale fields, and create a meaningful reuse window. Likewise, a second put/free on the same potentially dangling reference is a specific +1 lifetime mutation signal, not proof of arbitrary read/write. A constrained destructor UAF can remain Moderate even when RealUAF=YES.

KVM rmap, stale translation and synchronization

Shadow-page-table or reverse-map changes can leave stale relationships that later MMU traversals consume. The analysis distinguishes a missing zap/drop before overwrite, stale rmap traversal, active translation synchronization and an actual post-lifetime use. Host-controlled dirty logging or a VM-exit path alone does not establish a guest-triggerable host exploit; capability and actor boundaries must be traced end to end.

Remote unaligned access and calibration

If unauthenticated packet data directly reaches a typed, potentially unaligned kernel load and the fix replaces it with `get_unaligned_be32()`, the concrete remote fault-prone operation may earn +1 in addition to +2 remote reachability. A narrowly defined calibration may raise the final score to at least 4 when no independent negative constraint applies. This is a stated floor, not a fabricated “common networking” point or a claim that a panic was observed.

Cryptographic entropy integrity without memory corruption

An RDSEED/RDRAND instruction reporting success while returning demonstrably invalid or predictably biased entropy is a cryptographic security primitive even if no crash, UAF or OOB access exists. The methodology calls for manual review and typically AS 4–5, but does not assert weak cryptographic keys or Linux CRNG compromise without tracing actual consumers and mixing.

Persistent reference leak versus double-release

A missing `put()` that retains a resource is a concrete +1 leak, but eventual resource exhaustion alone remains a `CLASS_C`-style availability problem. A release followed by another release through a dangling reference is instead a lifetime/refcount issue and must be evaluated separately. Neither is automatically Important merely because repeated requests are possible.

10. What a useful AS explanation should publish

- The exact patch/commit and affected code path; attacker-controlled input and full trigger prerequisites.
- A signal ledger with each weight, source evidence and explicit non-awarded signals.
- Raw conservative/higher scores, any floor/cap, and the displayed value without hiding presentation differences.
- Primitive class, confidence in call-site analysis, uncertainty and separate manual-review/downgrade decisions.
- A short explanation of what evidence would change the assessment: a protected victim, a reachable consumer, namespace capability scope, controlled reclaim, or a demonstrated crash.

11. How the methodology was developed and stress-tested

ActionableScore did not emerge from a one-pass prompt asking an LLM to guess a number. It was developed through iterative, engineer-directed calibration of Linux kernel CVE assessments: review a real patch and the model's signal ledger, compare the outcome against a technical reading and other triage decisions, identify the exact false-positive or false-negative mechanism, add a narrowly scoped rule or counterexample, and rerun relevant regression cases. Alexander Larkin directed this work with LLM assistance. "Training" here means prompt engineering, example-based calibration, and regression testing—not gradient-based training or fine-tuning of the LLM's weights.

From compact heuristics to an evidence-constrained rubric

The current supplied scoring prompt is approximately 156 KB of text, 3,112 physical lines, and more than 20,000 whitespace-separated tokens/words (the latter is a rough text count, not a tokenizer measurement). Its size reflects accumulated edge-case rules and counterexamples: precise distinctions between reads and writes, active and retired objects, privilege namespaces and host privileges, reliable crash and theoretical DoS, ordinary memory corruption and ownership/COW violations, and whether an LLM's own earlier claims are admissible evidence. Length is not itself proof of accuracy; the rationale for the detail is to make ambiguous kernel mechanisms auditable and reduce repeated mistakes.

The calibration objectives pull in both directions. The system must recognize LOW-like issues and bounded crash-only behavior without treating every UAF, WARN, leak, or networking fix as IMPORTANT. At the same time it must not silently downgrade real attacker-influenced heap writes, live-object lifetime violations, shared-backing integrity changes, or security-boundary failures merely because no public exploit or complete LPE chain is available. Review decisions and severity-preservation decisions are deliberately distinct.

A repeatable calibration loop

- Select real kernel CVEs representing both ordinary low-impact bugs and high-risk memory-safety or boundary primitives; retain commit text, code changes, relevant call paths, and prior outputs.
- Run the assessment, recording positive and negative signal arithmetic, conservative/higher interpretations, primitive class, reachability assumptions, and review/downgrade flags.

- Inspect mismatches at the mechanism level: was a point invented from CVSS, was an unprivileged path assumed, was the same crash counted twice, or was a genuine write/ownership primitive dismissed as only DoS?
- Introduce a narrow correction with positive and negative examples, plus an explicit evidence threshold; avoid globally increasing all UAF or all network bugs.
- Rerun representative earlier cases to check both directions: LOW stays LOW where warranted, and IMPORTANT-grade mechanisms are not lost to aggressive downgrade rules. Preserve cases where the outcome remains uncertain for human review.

Examples from documented iterative testing

Regression theme	Documented CVE cases / observation	What was being checked
Low-end calibration	CVE-2025-38193; CVE-2022-50865; CVE-2023-53510	Tests included LOW classification and removing overly broad preservation or classifier gates. Ground-truth disputes were identified rather than hidden.
Important preservation	CVE-2026-72262; CVE-2025-40248	Explicit heap OOB write/read or an IMPORTANT-vs-MODERATE miss prompted scrutiny of automatic downgrades and evidence-preservation logic.
Crash vs stronger primitive	CVE-2026-97417; CVE-2022-49733; CVE-2026-98401	Remote direct-fault, crash/KPANIC handling, and stale transport-header OOB-read cases exercised whether crash priority is preserved without inventing LPE.
Score/reconciliation stability	CVE-2026-68320	Repeated runs yielded AS 6 versus 7 and different intermediate paths, while the final MODERATE 7.0 result remained stable; a reason to test repeatability and explicit decision rules.
Historical public examples	CVE-2026-53151 (AS 4); CVE-2026-80844 (AS 6)	Differentiate fragmented RXRPC ACK parsing from a concrete attacker-influenced AH6 out-of-bounds memmove, while documenting unresolved prerequisites.

The methodology has been iteratively exercised and manually reviewed against hundreds of Linux kernel CVE cases across the broader AL-KERNEL development process. Early regression cases were drawn from the original SimpleNet neural-network classifier’s dataset. Subsequent rounds used multiple changing CVE collections, supplemented by ongoing case-by-case checks as new and technically interesting vulnerabilities appeared—often at least one fresh case per day. These repeated reviews were used to refine signal definitions, add counterexamples, and test whether a change corrected the targeted error without reintroducing earlier false positives or false negatives. This is extensive engineering calibration, not a claim of a statistically independent benchmark or a measured ActionableScore accuracy rate.

Why the examples became unusually specific

A recurring failure mode in LLM kernel triage is turning familiar vocabulary into unsupported impact. The prompt therefore grew evidence firewalls and detailed counterexamples: a large OOB memcpy is not automatically a controlled overwrite; a dirty-fragment or Dirty-Pipe-like shared-page concern requires tracing which protected backing actually changes; an RCU/qdisc stale-object crash is not automatically exploitable UAF; and a remote typed unaligned load may deserve fault-specific attention even without a reproduced panic. Each example teaches a testable distinction rather than a memorized severity label.

Other refinements include explicitly handling cross-object live-state interference, KVM reverse-map and synchronization issues, checksum metadata consumed downstream, repeated release versus resource leak, and narrow preservation-only vetoes when upstream evidence supports unresolved memory safety. A preservation veto does not add score points or manufacture an exploit primitive; it prevents a premature automatic downgrade while a human resolves a concrete uncertainty.

What this work establishes—and what it does not

The development history supports substantial iterative prompt refinement, manual examination of hundreds of CVE cases across the project, and targeted attempts to reduce errors at both ends of the severity spectrum. It does not establish a formal prospective benchmark, a measured false-negative rate, or calibrated exploit probability. Published claims should distinguish observed test outcomes, engineering judgment, and future validation goals. A next step is a versioned public regression corpus with expected signal ledgers, independent human labels, repeated-run variance, and before/after false-positive and false-negative statistics.

12. Limitations and research direction

ActionableScore is preliminary, heuristic and not statistically calibrated to exploitation probability. Its point weights and exceptional floors are engineering choices subject to revision. LLMs can hallucinate call paths, overread CVSS, confuse namespaces, or count the same effect twice. Reports should preserve their evidence traces and accept human correction. Future work includes public per-signal explanations, labeled regression corpora, consistency checks, ablation studies, calibration against independently reviewed kernel CVEs, and explicit tracking of false-negative automatic downgrades.

References and methodology provenance

Project: <https://kernelcve.org/> | CVE-2026-53151: <https://kernelcve.org/list/?q=CVE-2026-53151> | CVE-2026-80844: <https://kernelcve.org/list/?q=CVE-2026-80844>

Methodology basis: supplied October 2026 ActionableScore scoring prompt and complete AL-KERNEL reports for both example CVEs. The two CVE signal breakdowns are historical recorded assessments, not independent validation of exploitation.