Your Scanner Says "Critical." FedRAMP Asks a Harder Question.

Matthew Venne
Chief Technology Officer, stackArmor

June 2026

A plain-language companion to "A Deterministic, CVSS–Environmental Method for FedRAMP Rev5 VDR/VER Vulnerability Prioritization."

The problem in one sentence

A vulnerability scanner tells you how bad a flaw is *in the abstract*; FedRAMP's new rules ask how bad it is *for the agency whose data sits on that specific box* — and those are not the same question.

A "Critical" CVE on a throwaway sandbox can matter less than a "Low" CVE on the database holding six agencies' records. Everyone in security knows this intuitively. The hard part is turning that intuition into a number you can defend in an audit, reproduce on every run, and explain to a 3PAO without hand-waving.

That's the gap this work fills.

What FedRAMP gave us, and what it left out

FedRAMP Rev5's Vulnerability Disclosure Report and Vulnerability Evaluation Requirements (VDR/VER) introduce a clean prioritization vocabulary:

- **PAIN** — *Potential Agency Impact*, rated N1 through N5. This is the "how bad for the agency" score.
- A **remediation matrix** — how many days (or hours) you have to fix something, based on its PAIN level, how exploitable it is, and whether it's reachable from the internet.

What FedRAMP deliberately *didn't* give us is the missing middle: the function that takes a specific CVE on a specific asset and spits out an N-level. They specified the output buckets and the deadlines, but not the classifier that fills the buckets. Every provider is left to invent that themselves.

An invented, analyst-by-analyst classifier is the worst of both worlds: it's not reproducible (two analysts disagree), and it's not defensible (you can't show your work). So the question becomes: **can we build that classifier in a principled, standardized way instead of making one up?**

The key insight: the classifier already exists

Here's the move at the heart of the proposal. **The thing FedRAMP asks for — "re-weight a vulnerability's impact by what's at stake on this asset" — is exactly what the CVSS Environmental metrics were built to do.**

CVSS (the scoring system behind those "Critical/High/Medium" labels) has a lesser-known *Environmental* group designed for precisely this: it lets the organization consuming a vulnerability re-score it based on how much *their* environment cares about confidentiality, integrity, and availability on the affected system. Those knobs are called CR, IR, and AR — the Confidentiality, Integrity, and Availability *Requirements*.

So the mapping is almost one-to-one:

What FedRAMP asks	What already answers it
How sensitive is this asset's data?	CVSS Security Requirements (CR / IR / AR)
How big is the potential agency impact?	CVSS Modified Impact Sub-Score
Is it likely to be exploited?	EPSS probability + KEV (known exploited)
Is it reachable from the internet?	Deployment/network context
One agency or many?	A scope multiplier on the result

The payoff: **you don't need to invent a risk algebra.** You assign each asset its CR/IR/AR, evaluate two facts about reachability and exploitation, and the published CVSS math does the rest. The output is deterministic — same inputs, same answer, every analyst, every run.

How it actually works (the no-math version)

Every vulnerability can hurt three things: **secrecy** (confidentiality), **correctness** (integrity), and **uptime** (availability). For each one, you multiply two numbers:

1. *How badly does this flaw break it?* (from the CVE's own CVSS vector)
2. *How much does this particular system depend on it?* (from the asset's criticality)

Combine those three into a single damage score from 0 to 1, then translate that score into a plain word: **Minimal, Narrow, Disruptive, or Debilitating.** Add one more fact — does this asset serve one agency or many? — and you land on an N-level from N1 to N5.

That's the whole engine. The *only* place human judgment enters the core scoring is three cut-points that decide where one word turns into the next — and those are meant to be documented, governed, and back-tested against how your senior analysts actually triage.

But where does an asset's criticality come from? Archetypes.

Everything above hinges on that second number — *how much does this system depend on secrecy, correctness, and uptime?* That's the CR/IR/AR. So the obvious question is: who decides that, and how do you keep it from becoming the same analyst-by-analyst guesswork the method was trying to escape?

The answer is **asset archetypes.** Instead of hand-rating every server in isolation, you sort each asset into a short catalog of roles based on **what it does and what data it touches** — a

database, a login service, a cache, a message bus, a CI/CD runner, a public load balancer, and so on. Each archetype comes with its CR/IR/AR criticality *pre-filled and justified*. Tag an asset with its archetype, and its criticality is inherited automatically.

The reason this matters isn't just convenience — it's the **chain of thought**. If you hand-score a host as "CR: Low, IR: High, AR: Low," six months later nobody remembers why. But if you tag it as a *data-backbone message broker*, the rationale is right there in the label, readily auditable, and consistent with every other broker in the estate.

One subtlety the method is opinionated about: **you classify by role, not by software.** The same in-memory store is one archetype when it's a throwaway cache (low criticality), a much higher one when it's holding session tokens (an identity/secrets role), and different again when it's acting as a job broker. *How it's used* drives the score, not *what it is*. The method even recommends a "control-plane lens first" rule: if an asset can deploy, orchestrate, hold cross-estate credentials, or actuate configuration, classify it by that powerful control function regardless of the data it happens to store.

So the full pipeline reads top to bottom like this:

THE WHOLE PIPELINE

Classify the asset (archetype) — what it does, what data it touches → **sets its CR/IR/AR** — the criticality multipliers for confidentiality, integrity, availability → **feeds the CVSS Environmental impact math** — re-weighting the CVE's raw impact by what's at stake *here* → **produces the PAIN level (N1–N5)** — plus, with the reachability and exploitation facts, a concrete deadline.

Two important guardrails on the archetype catalog:

- **It's an example, not a standard.** The proposal publishes a sample catalog as an *existence proof* — but every provider should derive its own from its actual FIPS-199 categorization, data inventory, and authorization boundary. Two providers holding different data can legitimately assign the same software different criticality.
- **Mislabeling is a downgrade attack.** Because the whole score rides on the tag, tagging an asset *down* makes its flaws look smaller than they are. So archetype tags should be controlled and reviewed like any other security setting — and unclassified assets fail loud at maximum criticality (see below), never quiet.

The same flaw, two very different verdicts

This is the part worth internalizing, because it's the entire point of the method. Take one identical remote-code-execution CVE:

- **On a crown-jewel datastore holding multiple agencies' data:** scores Debilitating, multi-agency → **N5**, remediate in hours.
- **On a single-agency throwaway sandbox:** same CVE, scores Disruptive → **N3**, a much longer clock.

The vulnerability didn't change. The *asset* did all the work. And a "Low"-severity availability flaw on a high-uptime message backbone can legitimately land at **N4 or N5** — decoupled entirely from the vendor's qualitative label.

The PAIN level says how *bad*. Two more facts say how *fast*.

The N-level tells you the potential impact, but not your deadline. Two real-world facts tighten the remediation clock:

- **Is it actually being exploited — or likely to be?** This is the *exploitability* axis. A flaw is treated as likely-exploitable if it clears an EPSS probability threshold (EPSS being the statistical "how likely is this to be exploited in the wild" score) **or** if it's known to be actively exploited — for example, listed in CISA's KEV catalog. Either signal is enough; whichever fires moves the finding to a faster remediation column.
- **Is it reachable from the internet?** A vulnerable surface an unauthenticated outside attacker can actually reach — through a public IP, a public load balancer, an ingress, or any edge path that doesn't authenticate first — is far more urgent than the same flaw buried behind authentication on an internal-only service. (A route gated by an identity-aware proxy in a *remote-access* role — your own people, with the application's own login still behind it — counts as *not* internet-reachable: the attacker can't get to the vulnerable code.)

Stack those up and the logic is intuitive: **more severe + more exploitable + more exposed = less time to fix.** A lookup table turns the combination into a concrete deadline, with tighter clocks for higher-assurance providers. So an internet-facing, actively-exploited flaw on a shared multi-agency edge might be a 12-hour fix, while the same-tier flaw that's sub-threshold on EPSS and tucked behind authentication gets a much longer runway.

There's also an optional refinement that can raise the *level* rather than the clock: where an authoritative source reports that exploiting a flaw grants *total* technical impact, the affected dimensions can be floored to High before scoring — pushing a weak-looking CVE up a tier. It never invents impact on a dimension the CVE doesn't touch.

Fail loud, not quiet

A critical design choice: **missing information never makes a problem look smaller.** An asset that hasn't been classified yet defaults to maximum criticality — so it scores loudly and surfaces itself for classification, rather than hiding in the noise. "Unknown" is treated as "serious until proven otherwise."

Scoring isn't the end — disposition is

A scanner hit isn't a confirmed problem. The flagged code might not be present, might never run, might need a configuration nobody set, or might already be neutralized by a control. Sorting that out is the **disposition** step, and it can only happen *after* a finding is detected.

The method records dispositions using **VEX** (Vulnerability Exploitability eXchange) — a standardized, signable, machine-readable artifact. Crucially, disposition is an *overlay, not a re-score*: a finding's computed PAIN is always retained on record even when it's suppressed. That makes every "this doesn't apply to us" decision **auditable, attributable, and reversible** — with stronger evidence required to wave off a high-stakes finding than a trivial internal one.

Why this matters for the bigger picture

FedRAMP's VDR/VER rule is its implementation of CISA's Binding Operational Directive 26-04 — the broader push to move remediation prioritization *off* raw CVSS base scores and *onto* exploitability and mission impact. FedRAMP kept the directive's philosophy but swapped in its own instruments (PAIN in place of SSVC, among others). Because it chose PAIN, an off-the-shelf SSVC tool doesn't satisfy the rule by itself — and the classifier that produces an N-level was exactly the missing piece. This method supplies it in standards-grounded terms.

The bottom line

You can summarize the entire proposal in four claims:

1. **FedRAMP specified the vocabulary and the deadlines but not the classifier.** Everyone has to build one.
2. **You don't have to invent it.** CVSS Environmental metrics already encode "re-weight impact by what's at stake" — which *is* the agency-impact question.
3. **With two human inputs per asset** — its archetype (which sets CR/IR/AR) and a one-agency-or-many flag — every finding gets a deterministic, auditable N-level and a concrete deadline. Everything else is derived.
4. **It's defensible.** Same inputs, same output, every time — with the full derivation attached to each finding, and a fail-safe that errs toward over-classifying rather than under.

The deeper paper has the worked examples, the exact formulas, the reference architectures, and a sample archetype catalog (offered as an example, *not* a standard — every provider should own its own mapping). But the idea you came for is simple: **stop asking "how bad is this CVE?" and start asking "how bad is this CVE *here*?" — and let a formula everyone can check answer it the same way every time.**

Read the full technical memo: A Deterministic, CVSS–Environmental Method for FedRAMP Rev5 VDR/VER Vulnerability Prioritization