Journal of Intelligent Systems and Internet of Things

♦ ISSN: 2690-6791 Vol: 19, No. 01, 2027 ♦ PP. 16–34

♦ Article DOI: https://doi.org/10.54216/JISIoT.190102



FULL LENGTH ARTICLE

Intelligent Early Warning for IoT Software Supply Chains: Self-Exciting Point-Process and Hierarchical Bayesian Modelling of Open-Source Security Advisories

Mahmoud A. Zaher[1,*]

[1] Associate Professor, Data Science Department, Faculty of Artificial Intelligence, Horus University (HUE), Egypt

Emails: mzaher@horus.edu.eg

Received: August 06, 2026 Revised: September 20, 2026 Accepted: October 30, 2026 ⋆ Corresponding author

ABSTRACT

Internet of Things gateways, edge controllers and smart-home hubs run on open-source packages, so the advisories published for those packages are the signal that intelligent vulnerability-monitoring systems must interpret. We model that signal as a stochastic process, using every reviewed advisory in the GitHub Advisory Database as of 31 July 2026 across eight package ecosystems. Binary segmentation places a regime change in February 2026, after which monthly volume rose almost fivefold, driven by maintainer-written advisories that now usually precede the National Vulnerability Database. A modulated Hawkes process, estimated by expectation–maximisation, separates batch release from triggering and finds that nearly half of advisories arrive in same-hour batches, while self-excitation between batches is weak (branching ratio 0.05, half-life eight hours). Risk is concentrated. Advisories per package are heavy-tailed (exponent 2.14), and a beta-binomial empirical Bayes model concentrates the 12% of unpatched entries in a small set of packages. IoT-related packages carry more Critical advisories (22% against 13%) and more unpatched entries (18% against 12%). An adaptive early-warning score derived from the fitted process anticipates follow-up disclosures within seven days with an AUC of 0.75, equal to the strongest heuristic and interpretable, although its probabilities run low during the surge.

Keywords: Internet of Things ▪ Intelligent systems ▪ Software supply chain ▪ Hawkes process ▪ Empirical Bayes

1. INTRODUCTION

Production software is mostly assembled from other people's packages, and Internet of Things (IoT) software is no exception. A home-automation hub runs Home Assistant on Python, an industrial gateway bridges OPC UA and MQTT through Node-RED on Node.js, and edge services written in Go or Rust sit between sensors and the cloud; each of the registries behind them is a public commons maintained largely by volunteers [1]. IoT fleets are slow and costly to patch. They have also been exploited at scale [2, 3, 4]. Their operators therefore increasingly hand the reading of security information to automated monitoring systems that must decide, without a human in the loop, which disclosures justify an over-the-air update. When one of those packages turns out to be vulnerable, the knowledge travels to downstream users through an advisory, a structured record that names the package, the affected version range and, when one exists, the first fixed release. Dependency scanners, software composition analysis tools and the software bills of materials encouraged by Executive Order 14028 [5] are only as current and as complete as the advisories they ingest [6].

The GitHub Advisory Database is the largest curated source. It publishes advisories in the Open Source Vulnerability

© 2027 The Author(s). Published by ASPG. This article is licensed under CC BY 4.0.

(OSV) schema [7] and mirrors them to a public Git repository. Review sets it apart. Curators check each record's package mapping and version ranges. Advisories enter by two main routes. Some are imported from the National Vulnerability Database (NVD) after a CVE has been published there; others are written by maintainers as repository security advisories, reviewed and published by GitHub as a CVE Numbering Authority, and only afterwards copied to NVD [8]. A third group is older CVEs, imported late. These routes differ in who writes the record, how fast it appears and how much it says about the fix.

The generating process deserves modelling now. NVD's enrichment backlog, which began in February 2024, left a large share of CVEs without NIST's product mappings or scores for months [9], and in April 2026 NIST announced that it would no longer routinely enrich CVEs outside a priority set [10]. Ecosystems are also attack targets [11, 12]. Then, in the first seven months of 2026, the number of reviewed advisories rose to more than twice the 2025 annual total, concentrated in a handful of fast-moving projects, and downstream tools that treat advisory arrival as a steady trickle, or that assume every advisory names a fix, inherited the change without notice. A count alone hides it.

Earlier empirical work on ecosystem vulnerabilities has asked how long flaws stay latent, how fast fixes ship and how slowly dependants adopt them [13, 14, 15, 16]. Most of it covers one ecosystem at a time, descriptively. Here we treat the advisory stream itself as the object of study and bring a set of statistical models to bear on it, organised around six research questions, each answered by its own model:

RQ1. When did advisory volume change regime, and where?

RQ2. Which advisories reach GitHub before NVD, and how long does GitHub's review take on each route?

RQ3. Do advisories for the same package excite one another in time once batch releases and the shared surge are accounted for?

RQ4. How concentrated are advisories across packages, and how much of the remediation gap (entries with no fixed version) sits in a few packages?

RQ5. Do ecosystems differ in reported weaknesses and severity labels?

RQ6. Can the fitted process drive an intelligent early-warning score, and how do IoT-related packages compare with the rest?

Four contributions follow from this design. The first is a reproducible snapshot, fixed to a Git commit of 31 July 2026, together with a normalisation procedure (Algorithm 1) that separates contemporaneous from retrospective advisories and groups same-hour releases into disclosure episodes. We then formulate a modulated Hawkes process with package-specific background rates and a shared exponential kernel, estimated by an expectation–maximisation (EM) algorithm, which separates batch publication from genuine follow-up disclosure even while total volume is surging. Ecosystem-level estimates complete the set (regime ratios, tail exponents, overdispersion of unpatched shares, Jensen–Shannon distances and severity agreement), each with its uncertainty, so that maintainers, registry operators and tool builders can compare ecosystems on one footing. Last, we turn the fitted process into an adaptive early-warning score for intelligent monitoring of IoT software stacks (Section 4.7) and test it on the 2026 surge, with a separate profile of packages built for IoT. Sections 2 to 6 cover related work, data, models, results and implications in turn.

2. BACKGROUND AND RELATED WORK

2.1 Advisory databases and disclosure routes

Coordinated vulnerability disclosure has long separated the moment a flaw is reported to a vendor from the moment it is made public, and the public record has historically been the CVE entry in NVD. For open-source packages that record is a poor fit. NVD describes products through Common Platform Enumeration identifiers that rarely match registry package names and often records versions coarsely, so software composition analysis tools keep their own mappings, and Imtiaz et al. [6] found that nine such tools reported substantially different vulnerability sets for the same applications. The OSV schema [7] answers part of the problem. It expresses affected versions as ranges of introduced, fixed and last-affected events in an ecosystem's own version syntax.

For GitHub's database, the largest public OSV source with human review, Segal et al. [8] analysed the review pipeline for 2019–2025 and found two paths: repository-advisory-first records were reviewed within about two days of the patch, while NVD-first records took a median of about 28 days even after GitHub automated its NVD imports in June 2022. Their queueing model attributes most of the delay to the NVD route, and our analysis extends that observation to July 2026, when the composition of the stream changed sharply, while adding models of arrival dynamics, concentration across packages and remediation coverage that a queueing view leaves out.

2.2 Vulnerabilities in package ecosystems

The npm registry has drawn the most attention, with studies that traced how vulnerable npm packages propagate through the dependency network and how long fixes take to reach dependants [13] and found that a few maintainer accounts could reach a large fraction of the registry, which makes npm a small world with high risks [17]. In PyPI, Alfadel et al. [14] reported a median of about three years between a flaw's introduction and its discovery and found that 40.86% of vulnerabilities were fixed only after public disclosure. In Maven, Yang-Smith and Abdellatif [18] linked CVE lifecycles to dependency adoption and found that dependants took a median of 151 days to adopt an available fix, even though most of the CVEs they studied had been patched upstream before public disclosure. How rarely developers update dependencies has been documented repeatedly [15, 19], Pashchenko et al. [20] argued that only dependencies actually reachable in deployed code should count, and Imtiaz et al. [16] studied how security releases are packaged and announced. Supply-chain attacks form a separate strand, with catalogues of real incidents and a taxonomy of attack vectors [11, 12].

That literature measures the life of a vulnerability inside one ecosystem, while the advisory stream across ecosystems, which is what downstream tools actually consume, has been described in counts but rarely modelled. Two gaps stand out. In particular, no study we found asks whether advisories

cluster in time beyond what a common trend explains, or estimates how unpatched advisories are distributed across packages once small-sample noise is removed.

2.3 Statistical models used

Three bodies of method are used below. Self-exciting point processes were introduced by Hawkes [21] and estimated by maximum likelihood by Ozaki [22] and Ogata [23]; Veen and Schoenberg [24] gave the EM algorithm that treats the unobserved branching structure as missing data, and the time-rescaling theorem [25] provides a goodness-of-fit test. Heavy-tailed count distributions are fitted with the method of Clauset et al. [26], with Vuong's likelihood-ratio test [27] for comparing non-nested alternatives, as implemented in the Python package of Alstott et al. [28], which handles discrete data and the choice of the lower cut-off. Overdispersed binomial proportions are handled with the beta-binomial model [29] and shrunk towards ecosystem means by empirical Bayes [30]; ecosystem weakness profiles are compared with the Jensen–Shannon divergence [31], whose square root is a metric [32], and grouped by average linkage [33]. Change points come from binary segmentation [34], a special case of the penalised searches reviewed by Killick et al. [35], and counts are modelled with negative binomial regression [36]. Severity agreement uses quadratic-weighted kappa [37] and the CVSS v3.1 and v4.0 specifications [38, 39].

3. DATA AND MEASUREMENT

3.1 Snapshot

GitHub publishes the database at github/advisory-database. We fixed the analysis at commit e432e45, made on 31 July 2026 at 22:37 UTC and the last commit before 1 August, and checked out the reviewed advisories only (the "github-reviewed" folder), leaving out the unreviewed records, most of which mirror NVD entries without any package mapping. The snapshot holds 34,050 reviewed advisories; 889 are marked withdrawn and are dropped, leaving 33,161 advisories with 62,091 affected-package entries across 13,060 packages. For severity comparisons we joined NVD records from the fkie-cad/nvd-json-data-feeds mirror at tag v2026.07.31-000014, and for exploitation status the CISA Known Exploited Vulnerabilities catalog at version 2026.07.29 [40]. Table 1 lists the sources.

Eight ecosystems account for 99.3% of advisories and are analysed individually: npm, PyPI, Maven, Go, Packagist (PHP), NuGet (.NET), crates.io (Rust) and RubyGems, while Hex, Pub, Swift and GitHub Actions together contribute 217 advisories and appear only in pooled totals.

3.2 Normalisation and disclosure episodes

Each advisory a carries a publication time p_a, a review time r_a and, when it has an NVD counterpart, the NVD publication time n_a. We define the signed lag to NVD and the review latency, both in days, as

$$l_a = p_a - n_a, \quad \rho_a = r_a - p_a, \tag{1}$$

so that $l_a < 0$ means GitHub published first. An advisory is a retrospective import if $l_a > 30$, that is, if GitHub published it more than a month after NVD; such records describe old vulnerabilities and would distort any analysis of arrival timing. All others are contemporaneous. An advisory is a maintainer advisory if its references include the repository security-advisory page on GitHub, which identifies a record drafted by the project itself; the remaining contemporaneous advisories with an NVD date are NVD-sourced.

Batch publication needs separate treatment. A maintainer who audits a project often publishes a dozen advisories within minutes, and a point-process model that saw each as a separate event would read the batch as explosive self-excitation. Algorithm 1 therefore merges advisories for the same package that are published within one hour into a disclosure episode $e = (t_e, m_e)$ with time t_e equal to the first publication and mark m_e equal to the number of advisories. For an affected entry with range events E, we call the entry unpatched when no event is of type "fixed",

$$U_{a,k} = 1\left[\nexists\, \varepsilon \in E_{a,k} : \mathrm{type}\,(\varepsilon) = \mathrm{fixed}\right], \tag{2}$$

and we record whether the first range starts at version zero, which in OSV means that every release before the fix is affected.

Algorithm 1 Advisory normalisation and episode construction

Require: advisory set A at commit c; window $[T_0, T_1)$; merge gap $h = 1$ hour

Ensure: cleaned set A^*, affected entries F, episodes E

$A^* \leftarrow \{a \in A : a \text{ not withdrawn}\}$; $F \leftarrow \emptyset$
for all $a \in A^*$ do
  $l_a \leftarrow p_a - n_a$; $\rho_a \leftarrow r_a - p_a$ ▷ Eq. (1)
  $R_a \leftarrow 1\,[l_a > 30]$ ▷ retrospective import
  $M_a \leftarrow 1\,[\text{repository-advisory URL} \in \text{refs}\,(a)]$ ▷ maintainer advisory
  for all affected entries k of a do
    $U_{a,k} \leftarrow 1\left[\text{no fixed event in} E_{a,k}\right]$ ▷ Eq. (2)
    $Z_{a,k} \leftarrow 1\,[\text{first introduced event} = 0]$
    $F \leftarrow F \cup \{(a, k, U_{a,k}, Z_{a,k})\}$
  end for
end for
$E \leftarrow \emptyset$
for all packages q do
  sort contemporaneous advisories of q with $p_a \in [T_0, T_1)$ by p_a
  open a new episode whenever the gap to the previous advisory exceeds h
  $E \leftarrow E \cup \{(q, t_e, m_e)\}$ ▷ t_e first time, m_e batch size
end for
return A^*, F, E

3.3 Weakness families and severity

CWE identifiers were grouped into eleven families that matter for package code (their shares by ecosystem appear in Section 5.6): cross-site scripting; SQL and query injection; command and code injection; deserialisation; path and file handling; resource exhaustion, including regular-expression denial of service; memory safety; access control and integrity, including prototype pollution and cross-site request forgery; server-side request forgery, open redirect and XML external entities; information exposure and weak cryptography; other; plus an "unspecified" class for records without a CWE. GitHub severity labels (Low, Moderate, High, Critical) come

Table 1. Data sources and snapshot identifiers

Source	Snapshot	Fields used	Role
GitHub Advisory Database, reviewed folder	Commit e432e45, 31 Jul 2026	Publication, review and NVD dates; aliases; severity; CVSS vectors; CWE; affected ranges; references	All analyses
NVD JSON mirror (fkie-cad)	Tag v2026.07.31-000014	NVD primary CVSS v3.x base score	Severity agreement (Section 5.7)
CISA KEV catalog	Version 2026.07.29	CVE identifiers, ransomware field	Exploitation overlap

from the advisory, and CVSS v3.1 base scores were recomputed from the stored vectors. Table 2 summarises the eight ecosystems.

IoT-related packages were identified with a curated lexicon of package names covering IoT protocols, platforms and device software: MQTT, CoAP, OPC UA, Modbus, Zigbee, Bluetooth and LoRaWAN libraries; Home Assistant, ESPHome, Node-RED, openHAB, ThingsBoard, OpenRemote, EdgeX Foundry and Apache IoTDB; the Frigate and motionEye camera platforms, the FUXA SCADA server, MicroPython modules, AWS IoT device SDKs and firmware tooling. We checked every match by hand and removed false positives, such as Jenkins plug-ins named after the Zephyr test tool and CocoaPods packages whose names contain "coap". The remaining set has 57 packages and 135 advisories. It misses IoT deployments of general-purpose libraries, so it profiles packages built for IoT; the IoT exposure of an ecosystem as a whole would need dependency data from real device images to estimate.

Note. Advisories are assigned to the ecosystem of their first affected package. Maintainer = maintainer advisories; Unpatched and From version 0 are shares of affected entries with no fixed version and with a range starting at version zero; KEV counts advisories whose CVE alias appears in the CISA catalog.

4. MODELS AND ALGORITHMS

4.1 Regime detection in monthly volume

Let N_t count contemporaneous advisories in month $t = 1,\dots,T$. The series runs from January 2019 to July 2026 ($T = 91$). For a segment $s:e$ the Poisson cost is the negative maximised log-likelihood,

$$C(N_{s:e}) = -\sum_{t=s}^{e}\left(N_t \log \hat{N}_{s:e} - \hat{N}_{s:e}\right), \quad \hat{N}_{s:e} = \frac{1}{e-s+1}\sum_{t=s}^{e} N_t, \tag{3}$$

and binary segmentation (Algorithm 2) splits a segment at κ whenever the gain

$$G(\kappa) = C(N_{s:e}) - C(N_{s:\kappa-1}) - C(N_{\kappa:e}) \tag{4}$$

exceeds a penalty λ. Because monthly counts are overdispersed, a Poisson penalty of $2\log T$ would split the series at every spike, and we therefore scale it by a local dispersion estimate $\hat{\varphi}$, the median variance-to-mean ratio over six-month windows, and use $\lambda = 2\hat{\varphi}\log T$, a quasi-likelihood analogue of the Bayesian information criterion.

Algorithm 2 Binary segmentation with dispersion-scaled penalty

Require: monthly counts $N_1,\dots,N_T$; minimum segment length $w = 4$
Ensure: set of change points K

```
φ̂ ← median_i Var(N_{i:i+5}) / Mean(N_{i:i+5})
λ ← 2 φ̂ log T; K ← {1, T+1}
repeat
    G* ← 0; κ* ← ∅
    for all adjacent pairs (s, e) in K do
        for κ = s + w to e − w do
            compute G(κ) from Eqs. (3)–(4)
            if G(κ) > G* then G* ← G(κ); κ* ← κ
        end for
    end for
    if G* > λ then K ← K ∪ {κ*}
until no split is added
return K \ {1, T+1}
```

The size of the change by ecosystem comes from a negative binomial regression on ecosystem–month counts N_{jt},

$$\log E[N_{jt}] = \beta_{0j} + \beta_1 \tau_t + (\gamma + \delta_j) D_t, \quad \mathrm{Var}(N_{jt}) = \mu_{jt} + \alpha\mu_{jt}^2, \tag{5}$$

where τ_t is time in years, D_t indicates months from January 2026, and $\exp(\gamma + \delta_j)$ is the 2026 regime ratio of ecosystem j net of the linear trend [36]. We choose the dispersion α by profile likelihood on a grid.

4.2 Disclosure ordering

For contemporaneous advisories with an NVD record, let $Y_a = 1[l_a < 0]$ indicate GitHub-first publication. We fit

$$\mathrm{logit}\Pr(Y_a = 1) = \eta_0 + \eta_1 \tau_a + \eta_2 M_a + \eta_{eco}^{\top} e_a + \eta_{sev}^{\top} s_a, \tag{6}$$

with M_a the maintainer-advisory indicator, e_a ecosystem dummies (npm as reference) and s_a severity dummies (Moderate as reference). Review latency ρ_a gets no parametric model. More than half of all values are exactly zero, and no parametric family fits that mass at zero together with the long right tail, so we summarise latency by origin with empirical quantiles.

Table 2. Reviewed advisories by ecosystem, 31 July 2026 snapshot

Ecosystem	Advisories	Packages	High or Critical (%)	Maintainer (%)	Retro-spective (%)	Unpatched (%)	From version 0 (%)	KEV
npm	6,546	3,482	59.5	54.6	7.0	18.5	60.2	11
PyPI	5,642	1,514	51.0	50.7	17.6	7.4	51.5	14
Maven	6,775	3,172	46.4	17.9	34.8	14.7	46.5	51
Go	4,390	1,405	47.8	64.0	9.3	10.9	56.9	7
Packagist	6,145	1,056	38.3	35.7	26.6	14.0	35.7	23
NuGet	1,016	621	58.1	56.5	33.0	1.9	59.1	13
crates.io	1,405	1,241	51.2	48.0	7.9	10.6	69.1	0
RubyGems	1,025	428	46.5	40.0	28.9	4.7	42.6	3

4.3 A modulated Hawkes process for disclosure episodes

Episodes of package q form a sequence $t_{q,1} < t_{q,2} < \ldots < t_{q,n_q}$ on the window $[0, T]$. It runs from 1 January 2023 to 31 July 2026 ($T = 1{,}308$ days). We model the conditional intensity as

$$\lambda_q(t \mid H_t) = \mu_q\, g(t) + \sum_{i:\, t_{q,i} < t} \alpha\beta\, e^{-\beta\left(t - t_{q,i}\right)}, \tag{7}$$

where μ_q is a package-specific background rate, $g(t)$ is a step function over calendar months shared by all packages and normalised so that $\int_0^T g(u)\, du = T$, α is the branching ratio (the expected number of episodes directly triggered by one episode) and β the decay rate of the exponential kernel [21]. The modulation g absorbs the common surge, so that α measures clustering within a package beyond the shared trend. Without g, the 2026 surge would pass for self-excitation. The log-likelihood is [23]

$$\log L = \sum_q \left[\sum_{i=1}^{n_q} \log \lambda_q\left(t_{q,i}\right) - \mu_q G(T) - \alpha \sum_{i=1}^{n_q} \left(1 - e^{-\beta\left(T - t_{q,i}\right)}\right) \right], \qquad G(t) = \int_0^t g(u)\, du. \tag{8}$$

Following Veen and Schoenberg [24], each episode is either an immigrant from the background or the offspring of one earlier episode. Given current parameters, the E-step computes the probabilities

$$\pi_{q,i0} = \frac{\mu_q g\left(t_{q,i}\right)}{\lambda_q\left(t_{q,i}\right)}, \qquad \pi_{q,ij} = \frac{\alpha\beta e^{-\beta\left(t_{q,i} - t_{q,j}\right)}}{\lambda_q\left(t_{q,i}\right)}, \qquad j < i, \tag{9}$$

and the M-step updates

$$\hat{\alpha} = \frac{\sum_q \sum_{i>j} \pi_{q,ij}}{\sum_q \sum_i \left(1 - e^{-\beta\left(T - t_{q,i}\right)}\right)}, \qquad \hat{\beta} = \frac{\sum_q \sum_{i>j} \pi_{q,ij}}{\sum_q \sum_{i>j} \pi_{q,ij}\left(t_{q,i} - t_{q,j}\right)}, \tag{10}$$

$$\hat{\mu}_q = \frac{\sum_i \pi_{q,i0}}{G(T)}, \qquad \hat{g}_m = \frac{\sum_q \sum_{i:\, t_{q,i} \in m} \pi_{q,i0}}{L_m \sum_q \hat{\mu}_q}, \tag{11}$$

with L_m the length of month m. The $\hat{\beta}$ update drops the boundary term in $\partial \log L / \partial \beta$, a standard approximation whose effect is negligible when $\beta T \gg 1$, and Algorithm 3 gives the full procedure, including the inner loop that alternates between the package rates μ_q and the monthly profile g. The nested model with $\alpha = 0$ is a modulated Poisson process, and the likelihood-ratio statistic between the two has two degrees of freedom; fit is checked by time rescaling [25], under which the compensator increments

$$z_{q,i} = \int_{t_{q,i-1}}^{t_{q,i}} \lambda_q(u)\, du \tag{12}$$

are independent unit exponentials. Because $\hat{\mu}_q$ is estimated from the same events, packages with one or two episodes give degenerate increments; the test is therefore applied to packages with at least five episodes. Uncertainty in $\left(\hat{\alpha}, \hat{\beta}\right)$ comes from 40 package-level bootstrap resamples.

Algorithm 3 EM estimation of the modulated shared-kernel Hawkes process

Require: episode times $\{t_{q,i}\}$; window $[0, T]$; monthly bins m of length L_m; tolerance $\varepsilon = 10^{-9}$

Ensure: $\hat{\alpha}, \hat{\beta}, \{\hat{\mu}_q\}, \{\hat{g}_m\}$

```
initialise α ← 0.2, β ← 0.5, g_m ← 1, μ_q ← n_q / T
repeat
  for all packages q and episodes i do ▷ E-step
    λ_{q,i} ← μ_q g(t_{q,i}) + Σ_{j<i} αβ e^{-β(t_{q,i} − t_{q,j})}
    compute π_{q,i0} and π_{q,ij} by Eq. (9)
  end for
  update α and β by Eq. (10) ▷ M-step
  for r = 1 to 3 do ▷ block coordinate ascent
    update μ_q and g_m by Eq. (11)
  end for
```

rescale g so that $\sum_m g_m L_m = T$ and recompute μ_q
evaluate $\log L$ by Eq. (8)
until relative change in $\log L < \varepsilon$
return $\hat{\alpha}, \hat{\beta}, \{\hat{\mu}_q\}, \{\hat{g}_m\}$

4.4 Heavy-tailed concentration

Let x_q be the number of advisories that name package q. For $x \geq x_{\min}$ the discrete power law has probability mass

$$\Pr(X = x) = \frac{x^{-a}}{\zeta(a, x_{\min})}, \qquad \zeta(a, x_{\min}) = \sum_{k=0}^{\infty} (k + x_{\min})^{-a}, \tag{13}$$

with ζ the Hurwitz zeta function. Following Clauset et al. [26], $\hat{a}$ is the maximum-likelihood estimate given $x_{\min}$, and $\hat{x}_{\min}$ minimises the Kolmogorov–Smirnov distance

$$D = \max_{x \geq x_{\min}} |S(x) - P(x)| \tag{14}$$

between the empirical and fitted tail distributions. A semi-parametric bootstrap with 100 synthetic data sets gives a goodness-of-fit p-value, and Vuong's normalised log-likelihood ratio R [27] compares the power law with the lognormal and exponential alternatives; $R > 0$ favours the power law. Inequality is also summarised by the Gini coefficient, and Algorithm 4 sets out the whole procedure, from the scan over candidate cut-offs to the bootstrap and the two Vuong comparisons.

Algorithm 4 Discrete power-law fit with bootstrap goodness of fit

Require: counts $x_1, \ldots, x_n$; bootstrap size $B = 100$
Ensure: $\hat{a}$, $\hat{x}_{\min}$, D_0, p_{gof}, $R_{\ln}$, $R_{\exp}$
for all candidate $x_{\min} \in \mathit{unique}(x)$ do
  $a(x_{\min}) \leftarrow \arg\max_a \sum_{x_i \geq x_{\min}} \log \Pr(X = x_i)$ ▷ Eq. (13)
  $D(x_{\min}) \leftarrow$ KS distance by Eq. (14)
end for
$\hat{x}_{\min} \leftarrow \mathit{argmin} D(x_{\min})$; $\hat{a} \leftarrow a(\hat{x}_{\min})$; $D_0 \leftarrow D(\hat{x}_{\min})$
$c \leftarrow 0$
for $b = 1$ to B do
  draw $n_t \sim \text{Binomial}(n, \#\{x_i \geq \hat{x}_{\min}\}/n)$
  resample $n - n_t$ values from $\{x_i < \hat{x}_{\min}\}$
  draw n_t values $\lfloor (\hat{x}_{\min} - \tfrac{1}{2})(1-u)^{-1/(\hat{a}-1)} + \tfrac{1}{2} \rfloor$, $u \sim U(0,1)$
  refit lines 1–5 to the synthetic sample to obtain D_b; $c \leftarrow c + 1[D_b \geq D_0]$
end for
$p_{gof} \leftarrow c/B$; compute Vuong ratios $R_{\ln}$, $R_{\exp}$
return $\hat{a}, \hat{x}_{\min}, D_0, p_{gof}, R_{\ln}, R_{\exp}$

4.5 Remediation coverage: logistic and empirical Bayes models

The probability that an affected entry is unpatched is modelled with a logistic regression on ecosystem, severity, weakness family, maintainer origin, retrospective import and publication time. Package-level variation is treated hierarchically. For package q in ecosystem j with n_q affected entries, of which k_q are unpatched, assume

$$k_q \mid \theta_q \sim \text{Binomial}(n_q, \theta_q), \quad \theta_q \sim \mathit{Beta}(a_j, b_j), \tag{15}$$

so that the marginal likelihood is beta-binomial [29],

$$\Pr(k_q) = \binom{n_q}{k_q} \frac{B(k_q + a_j, n_q - k_q + b_j)}{B(a_j, b_j)}, \tag{16}$$

with B the beta function. We estimate (a_j, b_j) by maximising the product of Equation 16 over packages with at least two entries (Nelder–Mead on the log scale), report the intra-package correlation $\rho_j = 1/(a_j + b_j + 1)$, and shrink each raw share towards the ecosystem mean with the posterior mean [30]

$$\tilde{\theta}_q = \frac{k_q + a_j}{n_q + a_j + b_j} = (1 - w_q)\frac{k_q}{n_q} + w_q \frac{a_j}{a_j + b_j}, \quad w_q = \frac{a_j + b_j}{n_q + a_j + b_j}. \tag{17}$$

A likelihood-ratio test against the plain binomial ($\rho_j = 0$) checks whether the hierarchy is needed at all, and Algorithm 5 lists the steps, from the per-ecosystem fit to the shrinkage weights w_q.

Algorithm 5 Beta-binomial empirical Bayes for package-level unpatched shares

Require: affected entries F with ecosystem j, package q and flag U
Ensure: $(\hat{a}_j, \hat{b}_j)$, $\hat{\rho}_j$, LR_j, posterior means $\tilde{\theta}_q$
for all ecosystems j do
  $P_j \leftarrow \{(k_q, n_q) : n_q \geq 2\}$ with $k_q = \sum U$, n_q = entries of q
  $(\hat{a}_j, \hat{b}_j) \leftarrow \mathit{argmax} \sum_q \log \Pr(k_q)$ by Eq. (16) ▷ Nelder–Mead on log scale
  $\hat{\rho}_j \leftarrow 1/(\hat{a}_j + \hat{b}_j + 1)$; $\hat{p}_j \leftarrow \sum k_q / \sum n_q$
  $LR_j \leftarrow 2\left[l_{\text{BB}}(\hat{a}_j, \hat{b}_j) - l_{\text{Bin}}(\hat{p}_j)\right]$
  for all $q \in P_j$ do
    $w_q \leftarrow (\hat{a}_j + \hat{b}_j)/(n_q + \hat{a}_j + \hat{b}_j)$; $\tilde{\theta}_q \leftarrow$ Eq. (17)
  end for
end for
return $\{\hat{a}_j, \hat{b}_j, \hat{\rho}_j, LR_j\}$, $\{\tilde{\theta}_q\}$

4.6 Weakness profiles and severity agreement

Let P_j be the distribution of weakness families in ecosystem j, excluding unspecified records. The Jensen–Shannon distance between ecosystems j and k is

$$d_{JS}(j,k) = \sqrt{\frac{1}{2} D_{\text{KL}}(P_j \parallel M) + \frac{1}{2} D_{\text{KL}}(P_k \parallel M)}, \quad M = \frac{1}{2}(P_j + P_k), \tag{18}$$

computed with base-2 logarithms so that it lies in $[0, 1]$ [31, 32]. Ecosystems are grouped by average linkage on d_{JS}, and cluster stability is the share of 300 bootstrap resamples (advisories resampled within each ecosystem) in which two ecosystems fall in the same of three clusters. A chi-square test of homogeneity and Cramér's V give an overall effect size.

For CVEs that have both a GitHub CVSS v3.1 vector and an NVD primary score, we compute the difference $\Delta = s_{NVD} - s_{GH}$, its Bland–Altman limits $\hat{\Delta} \pm 1.96\,\text{SD}(\Delta)$, the share of identical severity bands, and the quadratic-weighted kappa

[37]

$$\kappa_w = 1 - \frac{\sum_{u,v} w_{uv} O_{uv}}{\sum_{u,v} w_{uv} E_{uv}}, \quad w_{uv} = \frac{(u-v)^2}{(K-1)^2}, \tag{19}$$

where O and E are observed and chance-expected band proportions over $K = 4$ bands.

4.7 Adaptive early warning for intelligent monitoring

An automated monitoring agent for an IoT fleet sees advisories one at a time. After each new episode for a package it has to decide how likely another is within days, because a second disclosure soon after the first often means a further update to schedule. We derive such a score from the fitted process. The model is trained on January 2023 to December 2025, after which $\hat{\alpha}$, $\hat{\beta}$ and $\hat{g}$ are frozen, and the last training month's modulation $\hat{g}$ is carried forward. Package background rates are then updated online. A gamma prior Gamma(a_0, b_0), fitted by moments to the training estimates $\{\hat{\mu}_q\}$, is combined with the $N_q(t)$ episodes observed up to time t, of which $E_q(t) = \hat{\alpha} \sum_{t_{q,j} \le t} \left(1 - e^{-\hat{\beta}(t - t_{q,j})}\right)$ are expected offspring, giving the posterior mean

$$\widetilde{\mu}_q(t) = \frac{a_0 + N_q(t) - E_q(t)}{b_0 + G(t)}. \tag{20}$$

The early-warning score is the model probability of at least one further episode within Δ days,

$$s_q(t) = 1 - \exp\left\{ -\widetilde{\mu}_q(t)\, \hat{g}\Delta - \hat{\alpha} \sum_{t_{q,j} \le t} \left(e^{-\hat{\beta}(t - t_{q,j})} - e^{-\hat{\beta}(t + \Delta - t_{q,j})} \right) \right\}, \tag{21}$$

and Algorithm 6 gives the online procedure. With $\Delta = 7$ days, the score is evaluated on every episode published from 1 January to 24 July 2026, the label being whether the same package had another episode within seven days. Competing scores are the same model with background rates frozen at their training values (packages unseen in training receive the 10th percentile of $\hat{\mu}_q$), a modulated Poisson process in both versions, the number of earlier episodes of the package, and the time since its previous episode. Two criteria apply. Ranking quality is measured by the AUC and average precision with 500 bootstrap resamples, and operational value by precision and recall when the top 10% of scores trigger an alert, which over the whole database amounts to about 16 alerts a week in 2026.

Algorithm 6 Adaptive early-warning score for intelligent IoT vulnerability monitoring

Require: fitted $\hat{\alpha}$, $\hat{\beta}$, $\{\hat{\mu}_q\}$, $\{\hat{g}_m\}$ from Algorithm 3 on the training window $0 \le t < T_{\text{tr}}$; horizon $\Delta = 7$ days

Ensure: early-warning score $s_q(t)$ after each new episode

$(a_0, b_0) \leftarrow$ moment estimates of a gamma prior on $\{\hat{\mu}_q\}$
$\hat{g} \leftarrow \hat{g}_M$ ▷ last training month carried forward
for all new episodes of package q at time $t \ge T_{\text{tr}}$ do
  $N_q(t) \leftarrow$ episodes of q up to and including t
  $E_q(t) \leftarrow \hat{\alpha} \sum_{t_{q,j} \le t} \left(1 - e^{-\hat{\beta}(t - t_{q,j})}\right)$ ▷ offspring already realised
  $\widetilde{\mu}_q(t) \leftarrow \left(a_0 + N_q(t) - E_q(t)\right) / \left(b_0 + G(t)\right)$ ▷ Eq. (20)
  $\Lambda_q \leftarrow \widetilde{\mu}_q(t)\, \hat{g}\Delta + \hat{\alpha} \sum_{t_{q,j} \le t} \left(e^{-\hat{\beta}(t - t_{q,j})} - e^{-\hat{\beta}(t + \Delta - t_{q,j})}\right)$
  $s_q(t) \leftarrow 1 - e^{-\Lambda_q}$ ▷ Eq. (21)
end for
return $\{s_q(t)\}$

5. RESULTS

5.1 Volume and regime change (RQ1)

Annual counts conceal two different processes (Figure 1, panel a). The 2022 total of 8,656 advisories looks like a peak, but 58.9% of that year's records were retrospective imports of CVEs already in NVD, part of GitHub's expansion of ecosystem coverage. Contemporaneous volume grew far more steadily over the same years, from 731 advisories in 2019 to 3,572 in 2025, before the first seven months of 2026 produced 8,508, of which retrospective imports made up only 1.3% (Table 3). Maintainer advisories made up 8.2% of the 2019 total, 54.5% of 2025 and 84.1% of 2026.

Note. Withdrawn advisories excluded. The GitHub-first column covers contemporaneous advisories that also have an NVD publication date.

Algorithm 2, run on the 91 monthly totals of contemporaneous advisories with an estimated dispersion of $\hat{\varphi} = 13.4$ and penalty $\lambda = 120.8$, returned three change points (Figure 1, panel b), with mean monthly volume of 52.9 from January 2019 to August 2020, 147.1 from September 2020 to July 2021 and 280.9 from August 2021 to January 2026. From February 2026 it was 1,307.8, a 4.7-fold step. The two earlier breaks belong to the database's growth phase, when reviewed coverage was still being extended (Go advisories first appear in the monthly series in May 2021 and crates.io advisories in July 2021). The 2026 break is different in kind, since the surge it marks is produced by maintainers.

The negative binomial model of Equation 5 attributes that step to most ecosystems, though unevenly (Table 4, Figure 2). Net of a background trend of 34% growth a year (incidence-rate ratio 1.34, 95% CI 1.29–1.38), monthly volume in 2026 rose by a factor of 4.0 in NuGet, 3.8 in Go, 2.5 in PyPI and 2.4 in npm, with confidence intervals excluding one. Maven did not move (0.89, 0.44–1.78), and Packagist, RubyGems and crates.io had wide intervals that include one. Package concentration accounts for much of the surge. The twenty packages with the most 2026 advisories supplied 21.5% of that year's contemporaneous total; openclaw, the npm package of the OpenClaw AI-agent gateway, alone accounted for 590, and the next largest were Magick.NET (143 per package variant), the AVideo platform (132), n8n (122) and Open WebUI (95).

Note. Regime ratios are $\exp(\gamma + \delta_j)$ from Equation 5 with negative binomial dispersion $\hat{\alpha} = 0.775$ and a common trend of 1.34 per year. Change points from Algorithm 2: September 2020, August 2021 and February 2026.

5.2 Disclosure ordering and review latency (RQ2)

Which source publishes first has reversed. Among contemporaneous advisories with an NVD record, the share published on GitHub before NVD rose from 18.0% in 2023 to 51.9% in

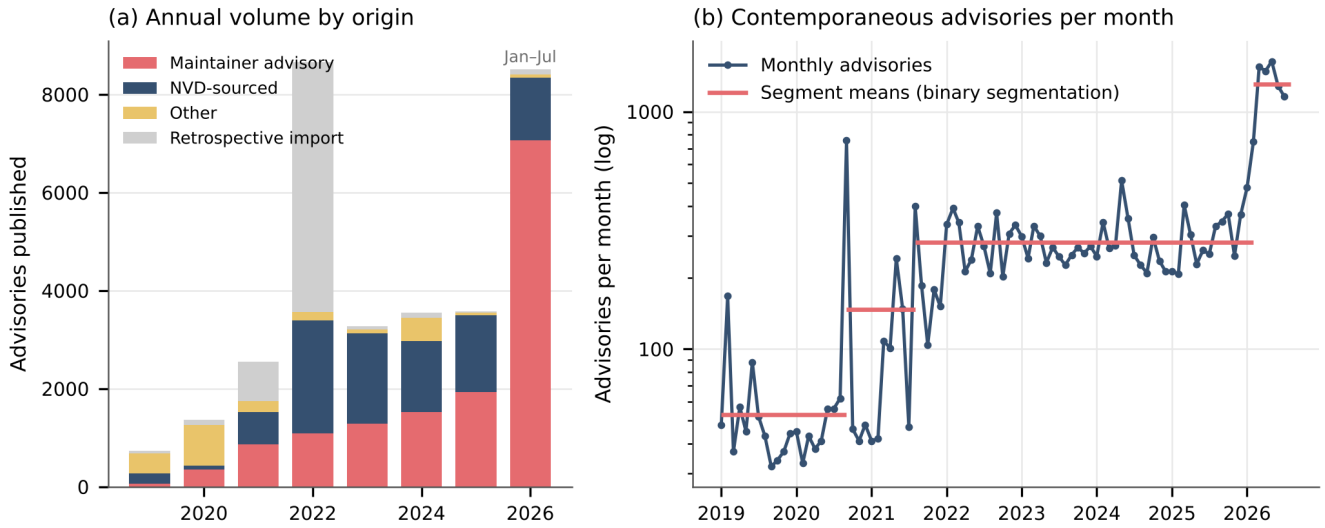


Figure 1. Advisory volume. (a) Reviewed advisories by publication year and origin; retrospective imports are advisories published more than 30 days after the corresponding NVD record; 2026 covers January to July. (b) Contemporaneous advisories per month on a log scale, with segment means from binary segmentation (Algorithm 2).

Table 3. Composition of reviewed advisories by publication year

Year	Advisories	Retrospective (%)	Maintainer advisories (%)	GitHub-first among contemporaneous with NVD record (%)	Critical (%)	CVSS v4.0 vector present (%)
2019	731	6.4	8.2	5.8	15.7	12.9
2020	1,371	7.8	26.6	50.7	37.4	9.3
2021	2,548	31.5	36.7	13.0	17.0	16.6
2022	8,656	58.9	13.1	10.2	12.2	10.8
2023	3,270	2.1	40.2	18.0	15.1	7.7
2024	3,546	3.0	44.4	23.2	10.2	39.8
2025	3,572	0.5	54.5	26.3	10.0	41.0
2026 (Jan–Jul)	8,508	1.3	84.1	51.9	10.9	40.4

Table 4. Change points and 2026 regime ratios by ecosystem

Ecosystem	Contemporaneous advisories 2025	Contemporaneous advisories 2026 (Jan–Jul)	2026 regime ratio	95% CI
npm	590	2,414	2.41	1.20–4.82
PyPI	678	1,595	2.49	1.24–4.98
Maven	691	694	0.89	0.44–1.78
Go	672	1,613	3.76	1.87–7.52
Packagist	571	1,134	1.66	0.83–3.33
NuGet	92	306	4.03	1.99–8.19
crates.io	176	396	2.01	1.00–4.05
RubyGems	58	173	1.45	0.71–2.96

2026 (Figure 3, panel a). The origin of the advisory decides most of this. Maintainer advisories reached GitHub first in 54.5% of cases, with a median lead of about 38 minutes, because GitHub as CNA publishes the CVE and its own record together and NVD ingests it afterwards; NVD-sourced advisories reached GitHub first in 1.2% of cases, with a median delay of 2.2 hours. Panel b of Figure 3 plots this as two separated distributions, with NVD-sourced records concentrated between one and three hours after NVD, consistent with an automated import that runs shortly after NVD publication, and maintainer advisories spread on both sides of zero.

Table 5 gives the logistic model of Equation 6. Maintainer origin multiplies the odds of GitHub-first publication by 88.4 (95% CI 72.6–107.7), and each year multiplies them by 1.30 even after origin is held fixed. Ecosystems differ as well. RubyGems advisories were 2.7 times as likely as npm advisories to appear on GitHub first, and NuGet advisories less than a third as likely; origin alone does not explain that gap,

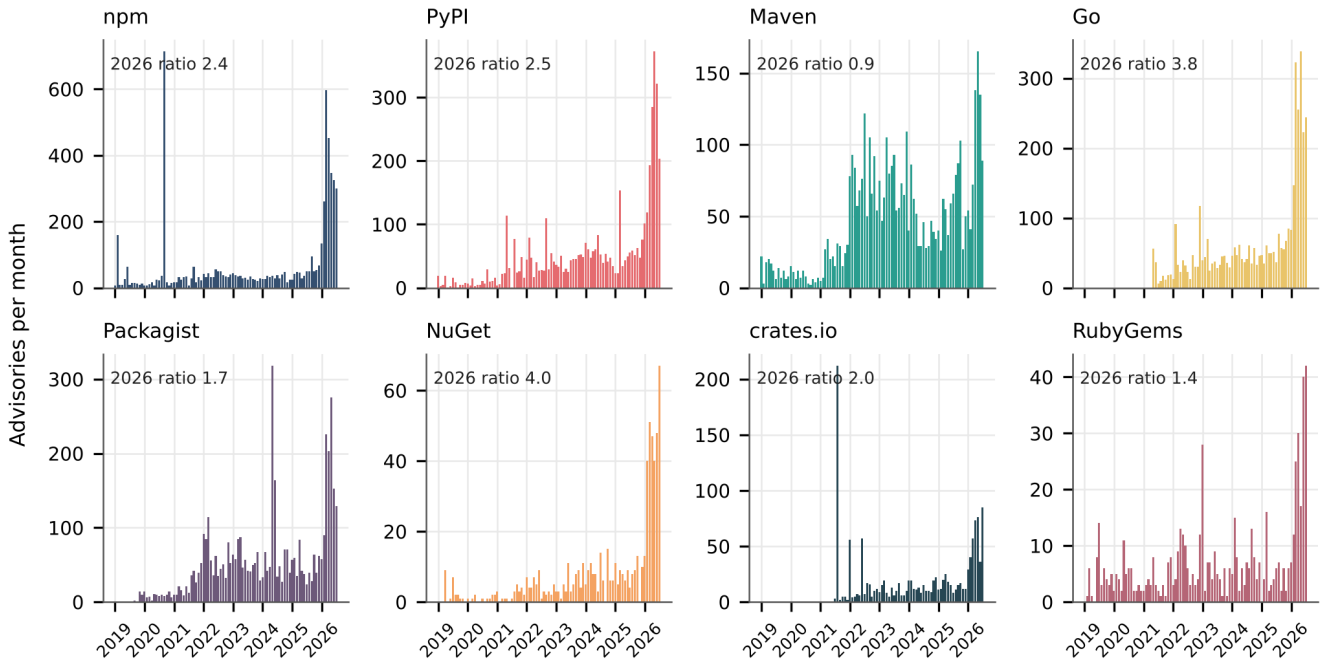


Figure 2. Contemporaneous advisories per month in the eight ecosystems, January 2019 to July 2026, with the 2026 regime ratio from the negative binomial model.

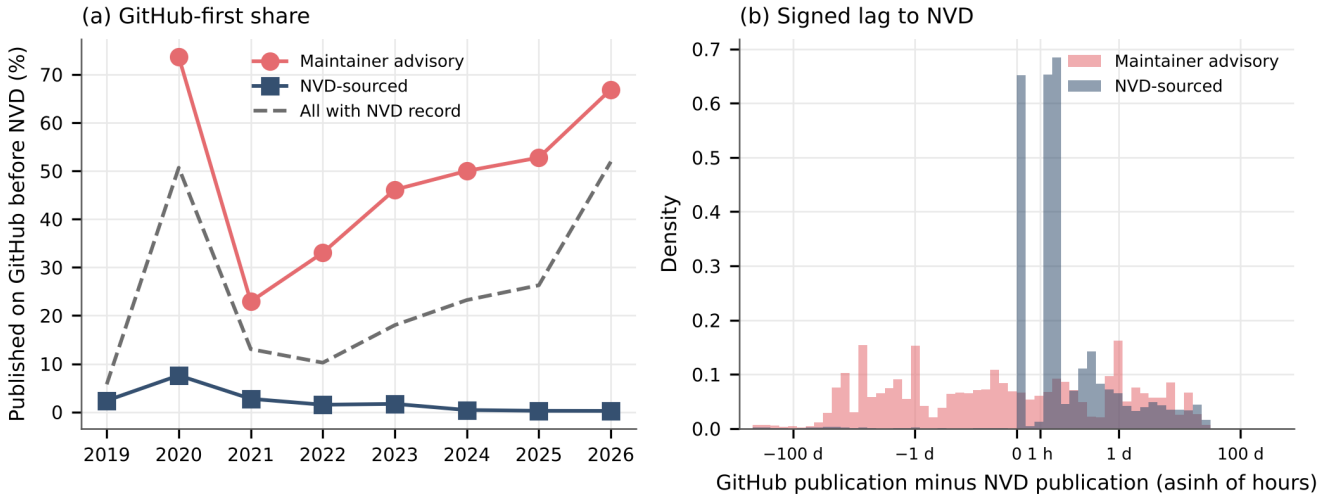


Figure 3. Disclosure ordering relative to NVD for contemporaneous advisories. (a) Share published on GitHub before NVD by year and origin (years with fewer than 20 advisories in a group omitted). (b) Distribution of the signed lag l_a on an inverse-hyperbolic-sine scale of hours.

and project-level data on who adopts repository advisories for coordinated releases would be needed to settle it. Severity had almost no effect (Critical 1.16, 1.01–1.33). The model explains a large share of the variation (McFadden pseudo-$R^2 = 0.39$).

Note. 19,273 advisories; McFadden pseudo-$R^2 = 0.388$.

Review latency follows the same split (Figure 4). Among contemporaneous advisories since 2019, 53.9% were reviewed within the same minute as publication, which happens when GitHub's curators publish a repository advisory as already reviewed; the 75th percentile was 0.39 days and the 95th 35 days. For maintainer advisories, 90.8% were reviewed in the same minute. NVD-sourced advisories took a median of 1.0 day and a 90th percentile of 35 days, while the small "other" group, mostly advisories with no NVD record, had a 90th percentile of 449 days. In 2026, 84.8% of all reviewed advisories were reviewed at publication. These results are not directly comparable to the review pipeline studied by Segal et al. [8]: our one-day value is the median time from GitHub publication to review for contemporaneous NVD-sourced advisories, excluding retrospective imports.

5.3 Batches and self-excitation (RQ3)

From January 2023 to July 2026, the 18,601 contemporaneous advisories with a primary package formed 12,297 disclosure episodes across 6,176 packages. Batches dominate. Only 19.7% of episodes contained more than one advisory, yet those episodes held 46.9% of the advisories, and the episode-size distribution falls roughly as a power of size (Figure 5, panel a). The largest single episode released 68 advisories for openclaw within one hour, and Gitea released 50 in another. Packagist had the highest batch share (60.7%), Maven and

Table 5. Logistic model of GitHub-first publication, contemporaneous advisories with an NVD record, 2019–2026: odds ratios

Covariate	Odds ratio	95% CI	p
Maintainer advisory	88.44	72.64–107.70	< 0.001
Publication time (per year)	1.30	1.27–1.33	< 0.001
Go (vs npm)	1.17	1.03–1.32	0.014
Maven	0.92	0.79–1.08	0.304
NuGet	0.30	0.24–0.38	< 0.001
Packagist	0.87	0.76–0.99	0.035
PyPI	0.81	0.72–0.91	0.001
RubyGems	2.71	2.05–3.60	< 0.001
crates.io	0.93	0.75–1.16	0.509
Critical (vs Moderate)	1.16	1.01–1.33	0.034
High	1.04	0.95–1.14	0.404
Low	0.93	0.80–1.09	0.380

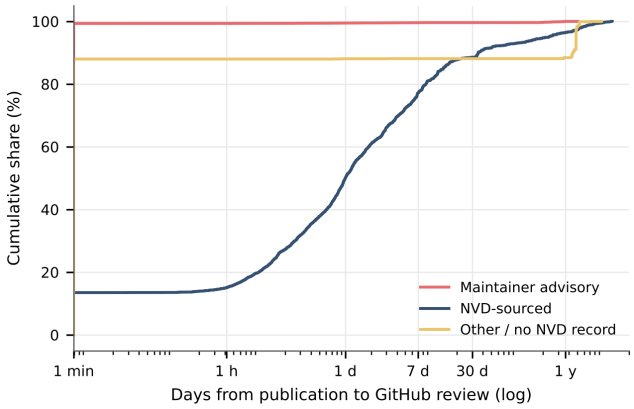


Figure 4. Empirical distribution of GitHub review latency ρ_a by origin, contemporaneous advisories published 2019–2026, on a log scale from one minute to five years.

crates.io the lowest (33.6% each).

Once batches are collapsed, how much clustering remains? Table 6 and Figure 6 give the answer from Algorithm 3. Pooled over all packages, the branching ratio is $\hat{\alpha} = 0.047$ (bootstrap 95% CI 0.041–0.052): about one episode in 21 is triggered by an earlier episode of the same package, and the remaining 95% arise from the background. The kernel is short. With $\hat{\beta} = 2.05$ per day (1.73–2.51), the half-life $\log 2/\hat{\beta}$ is 8.1 hours, which describes a maintainer releasing a batch in two or three bursts over an afternoon far better than it describes independent researchers finding related flaws weeks later. Self-excitation was strongest in Packagist ($\hat{\alpha} = 0.086$) and weakest in crates.io (0.023). All likelihood-ratio statistics against the modulated Poisson process are large (pooled 1,490 on two parameters), but β is not identified under $\alpha = 0$, so the usual chi-square reference is only approximate; the Akaike differences, all below -19, point the same way.

Note. Package-level bootstrap (40 resamples) for the pooled model: α 0.041–0.052, β 1.73–2.51. LR is twice the log-likelihood difference.

The background modulation carries the surge (Figure 6, panel a). It averaged 0.79 during 2025, rose to 1.27 in January 2026 and peaked at 3.25 in May. A model without it would have attributed that rise to triggering: an unmodulated version of Equation 7 with constant μ_q, fitted to the same data, returned $\hat{\alpha} = 0.062$. Fit is imperfect and we report it plainly. On the 436 packages with at least five episodes (4,515 intervals), the Kolmogorov–Smirnov distance between the rescaled intervals and the unit exponential fell from 0.155 for the modulated Poisson process to 0.139 for the Hawkes model, and the coefficient of variation of the intervals from 1.52 to 1.36, against 1.00 under a correct model. The residual overdispersion (panel c) means that some packages alternate quiet years with active months in ways that neither a month-level shared profile nor an eight-hour kernel captures; a package-specific slowly varying background, or a second, slower kernel, would be the next model to try. A cruder measure agrees with the small branching ratio. Of the 11,773 episodes with at least 30 days of follow-up, 20.8% were followed by another episode for the same package within 30 days.

5.4 Concentration across packages (RQ4)

Advisories are spread very unevenly over packages (Figure 7, Table 7). Of the 13,060 packages named in reviewed advisories, 65.4% appear once; the top 1% of packages hold 31.6% of all package–advisory pairs, and the pooled Gini coefficient is 0.63. Algorithm 4 selects $\hat{x}_{\min} = 1$ for the pooled data and estimates a tail exponent of $\hat{a} = 2.14$ (standard error 0.010), so the whole distribution, and not only its upper tail, is in the fitted range. The bootstrap goodness-of-fit p-value of 0.96 gives no reason to reject the power law, but a log-normal fits equally well ($R = -0.09$, $p = 0.93$), so the data cannot say which of the two heavy tails is the true one; both decisively beat the exponential ($R = 15.6$).

Note. Positive Vuong ratios favour the power law. Pooled bootstrap goodness-of-fit p = 0.96 (100 resamples).

Ecosystems separate along the tail. Exponents below two, which imply an infinite mean in the limiting distribution and in practice a few packages that dominate their ecosystem's record, occur in PyPI (1.96), Go (1.92), Packagist (1.83) and NuGet (1.83). In NuGet the power law beats the lognormal ($R = 3.40$), a consequence of a few native-library wrappers (Magick.NET and ChakraCore builds) that inherit every advisory of the upstream C or C++ code; in Go and Packagist the lognormal is preferred. crates.io is the least concentrated ecosystem (Gini 0.34, exponent 2.73), with no package above

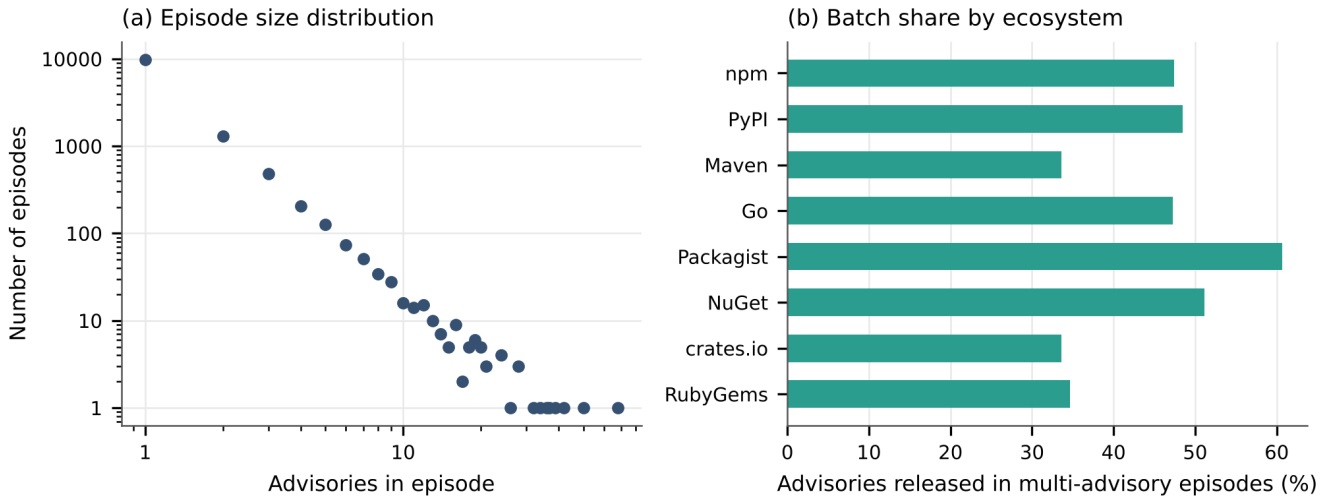


Figure 5. Disclosure episodes, January 2023 to July 2026. (a) Number of episodes by size (advisories for the same package published within one hour), log–log axes. (b) Share of each ecosystem's advisories released in episodes of two or more.

Table 6. Modulated Hawkes estimates for disclosure episodes, January 2023 – July 2026

Ecosystem	Packages	Episodes	Branching ratio α	Decay β (per day)	Half-life (hours)	LR vs modulated Poisson
All eight	6,176	12,297	0.047	2.05	8.1	1,490.1
npm	1,300	2,395	0.050	1.64	10.1	237.6
PyPI	946	2,248	0.042	2.17	7.7	246.3
Maven	1,376	2,201	0.033	1.99	8.3	182.6
Go	939	2,220	0.039	3.56	4.7	239.9
Packagist	685	1,804	0.086	1.61	10.4	446.5
NuGet	195	327	0.044	2.64	6.3	23.5
crates.io	473	674	0.023	2.58	6.5	43.2
RubyGems	160	301	0.033	7.25	2.3	36.3

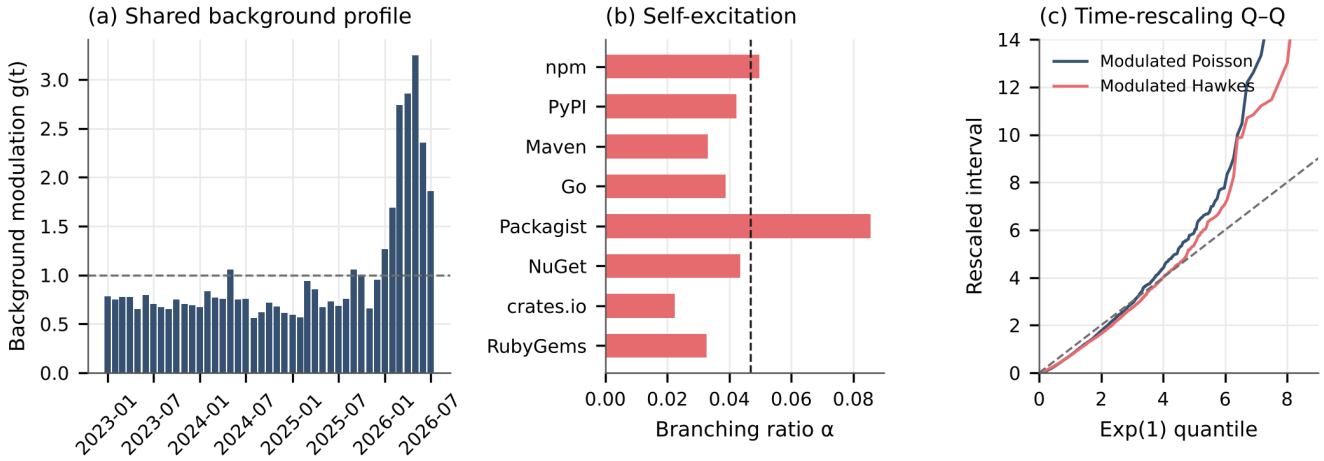


Figure 6. Modulated Hawkes process. (a) Estimated monthly background modulation $\hat{g}(t)$, normalised to average one. (b) Branching ratio by ecosystem; the dashed line marks the pooled estimate. (c) Quantile–quantile plot of rescaled inter-episode intervals against the unit exponential, packages with at least five episodes.

52 advisories.

5.5 Remediation coverage (RQ4)

Of 61,813 affected entries in the eight ecosystems, 11.7% list no fixed version. The share ranges from 1.9% in NuGet to 18.5% in npm (Table 2) and moves over time without a clean trend (Figure 8). npm's share fell from 31% in 2019 to 7% in 2026, while Maven's and Go's rose to 17% and 15% in 2026. The OSV range syntax adds a second dimension. Between 35.7% (Packagist) and 69.1% (crates.io) of entries declare every version up to the fix as affected, which is often a curator's safe default when the introducing commit is unknown.

Table 7. Heavy-tail fits for advisories per package

Ecosystem	Packages	Mean	Max	Gini	Cut-off	Exponent a (SE)	Vuong R vs lognormal (p)	Vuong R vs exponential (p)
All	13,060	3.27	590	0.63	1	2.14 (0.01)	−0.09 (0.93)	15.63 (< 0.001)
npm	3,482	2.08	590	0.48	1	2.52 (0.03)	−1.70 (0.09)	3.94 (< 0.001)
PyPI	1,514	4.49	433	0.69	1	1.96 (0.02)	−1.64 (0.10)	5.74 (< 0.001)
Maven	3,172	2.59	250	0.52	4	2.53 (0.07)	0.22 (0.82)	3.98 (< 0.001)
Go	1,405	3.75	199	0.62	1	1.92 (0.02)	−3.30 (0.001)	6.26 (< 0.001)
Packagist	1,056	6.58	437	0.75	1	1.83 (0.03)	−2.77 (0.006)	8.73 (< 0.001)
NuGet	621	7.97	247	0.79	1	1.83 (0.03)	3.40 (< 0.001)	10.19 (< 0.001)
crates.io	1,241	1.60	52	0.34	1	2.73 (0.05)	−0.54 (0.59)	4.41 (< 0.001)
RubyGems	428	2.63	62	0.54	1	2.16 (0.06)	−1.65 (0.10)	4.47 (< 0.001)

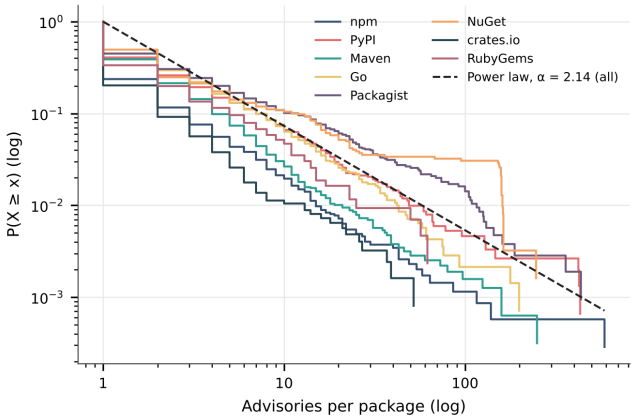


Figure 7. Complementary cumulative distribution of the number of advisories per package by ecosystem, log–log axes, with the pooled power-law fit.

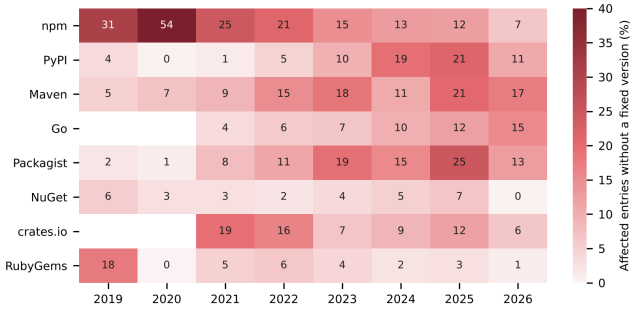


Figure 8. Share of affected entries with no fixed version, by ecosystem and publication year (2019–2026).

The logistic model in Figure 9 separates these influences. The strongest is origin. An entry in a maintainer advisory has 0.14 times the odds (95% CI 0.13–0.15) of being unpatched, because maintainers usually publish the advisory together with the release that fixes it, and retrospective imports have about half the odds (0.50), since old CVEs have had time to be fixed. Critical advisories are more often unpatched (1.88, 1.73–2.04), as are path and file-handling flaws (1.33) and cross-site scripting (1.16); memory-safety, information-exposure and resource-exhaustion flaws are less often unpatched (0.63, 0.70, 0.77). Every other ecosystem has lower odds than npm, from 0.58 for crates.io to 0.20 for NuGet and RubyGems. Holding origin fixed, the odds rise by 11% a year, so the falling raw share in 2026 is a composition effect of the maintainer surge and not an improvement within any route.

Package-level shares are heavily overdispersed (Table 8): the beta-binomial fit gives intra-package correlations $\hat{\rho}_j$ from 0.26 in NuGet to 0.75 in crates.io, every likelihood-ratio test against the plain binomial exceeds 64, and the binomial model, which treats every entry in an ecosystem as an independent draw, is rejected everywhere. In words, whether one entry of a package is unpatched predicts strongly whether its other entries are. The prior means $a_j/(a_j+b_j)$ sit close to the raw ecosystem shares. Their shapes do not. The priors are U-shaped ($a_j < 1$, $b_j < 1$ in six ecosystems), which places most packages near zero and a minority near one. Figure 10 plots the resulting shrinkage. Packages with two or three entries move most, with a median weight on the prior w_q between 0.14 (RubyGems) and 0.49 (NuGet), while large packages keep their raw share. After shrinkage, 213 Maven packages, 80 npm packages and 75 Go packages have posterior unpatched probabilities above one half.

The top of the posterior ranking turns up a data-quality problem alongside genuine gaps. Magento's project-community-edition package has 160 unpatched entries in 161, but these are advisories whose fix ships in a differently named Magento package, so the "unpatched" flag here describes the

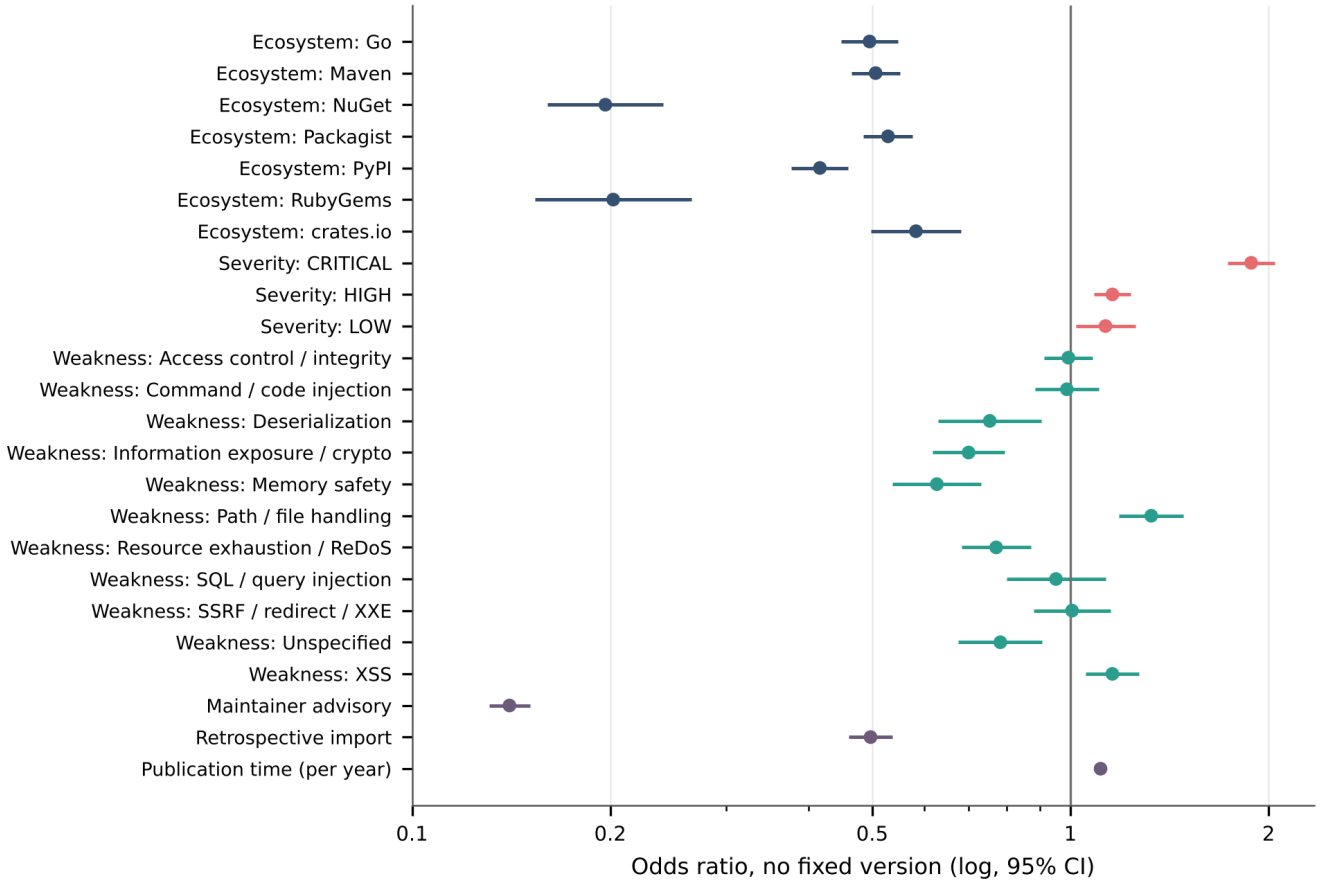


Figure 9. Logistic model of an affected entry having no fixed version, published 2019–2026: odds ratios with 95% confidence intervals on a log scale. References are npm, Moderate severity and the "other" weakness family. McFadden pseudo-R^2 = 0.131.

Table 8. Beta-binomial empirical Bayes model of unpatched entries by package

Ecosystem	Packages (≥ 2 entries)	a	b	Prior mean	ρ	LR vs binomial	Median w	Posterior $>$ 0.5 (packages)
npm	1,028	0.097	0.732	0.117	0.55	939.0	0.22	80
PyPI	676	0.143	0.906	0.136	0.49	1,592.8	0.21	65
Maven	1,587	0.089	0.453	0.164	0.65	2,322.1	0.15	213
Go	701	0.231	1.206	0.161	0.41	919.0	0.32	75
Packagist	622	0.130	0.834	0.135	0.51	3,764.9	0.19	63
NuGet	362	0.094	2.781	0.033	0.26	110.2	0.49	5
crates.io	290	0.050	0.277	0.153	0.75	344.5	0.14	39
RubyGems	173	0.027	0.598	0.043	0.62	64.0	0.14	5

range mapping and not the absence of a release. Others look genuine: funadmin (26 of 26) and the PowerJob and Atomix Java projects (7 of 7 each) appear to have advisories without any fixed release.

5.6 Weakness profiles (RQ5)

The eight ecosystems report different weaknesses (Figure 11). Cross-site scripting makes up 33% of classified Packagist advisories and 12% or less elsewhere; memory-safety flaws make up 38% in NuGet and 30% in crates.io, against 1% in npm and Maven; resource exhaustion, including regular-expression denial of service, is the second-largest family in PyPI and Go. A chi-square test of homogeneity rejects equal profiles ($\chi^2 = 8,894$ on 70 degrees of freedom), with a moderate effect size (Cramér's $V = 0.20$).

Jensen–Shannon distances (Equation 18, Figure 12) range from 0.16 between PyPI and RubyGems to 0.63 between Packagist and crates.io. Average linkage produces three groups: the five general-purpose ecosystems (npm, PyPI, Maven, Go and RubyGems), Packagist on its own, and NuGet with crates.io, the two whose advisories most often concern native code. The grouping is stable. In all 300 bootstrap resamples the same three clusters formed. The median half-width of the bootstrap intervals for the pairwise distances was 0.023, an order of magnitude below the smallest gap between clusters.

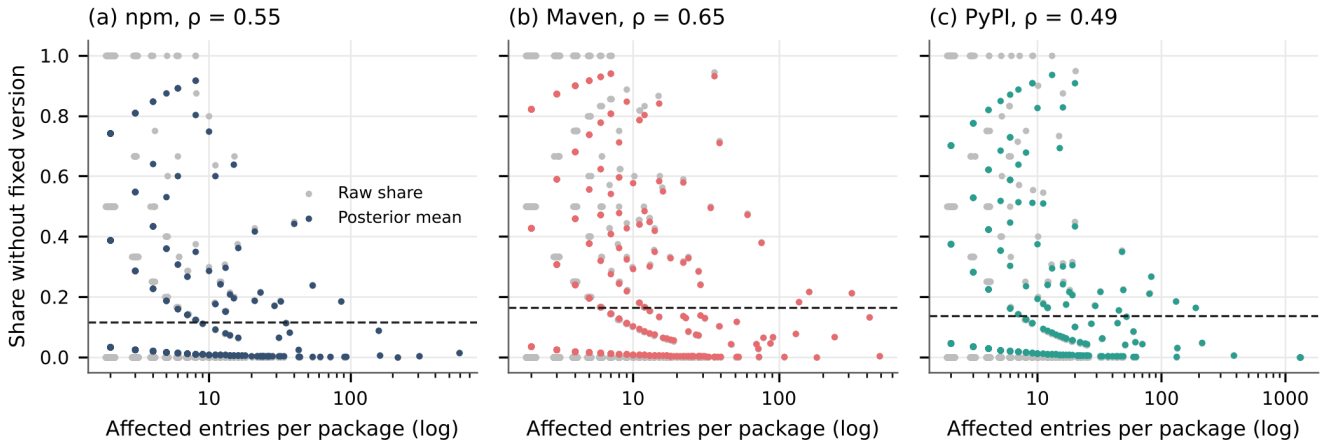


Figure 10. Empirical Bayes shrinkage of package-level unpatched shares in npm, Maven and PyPI. Grey points are raw shares k_q/n_q, coloured points posterior means $\widetilde{\theta}_q$ (Equation 17), against the number of affected entries; dashed lines mark the prior mean.

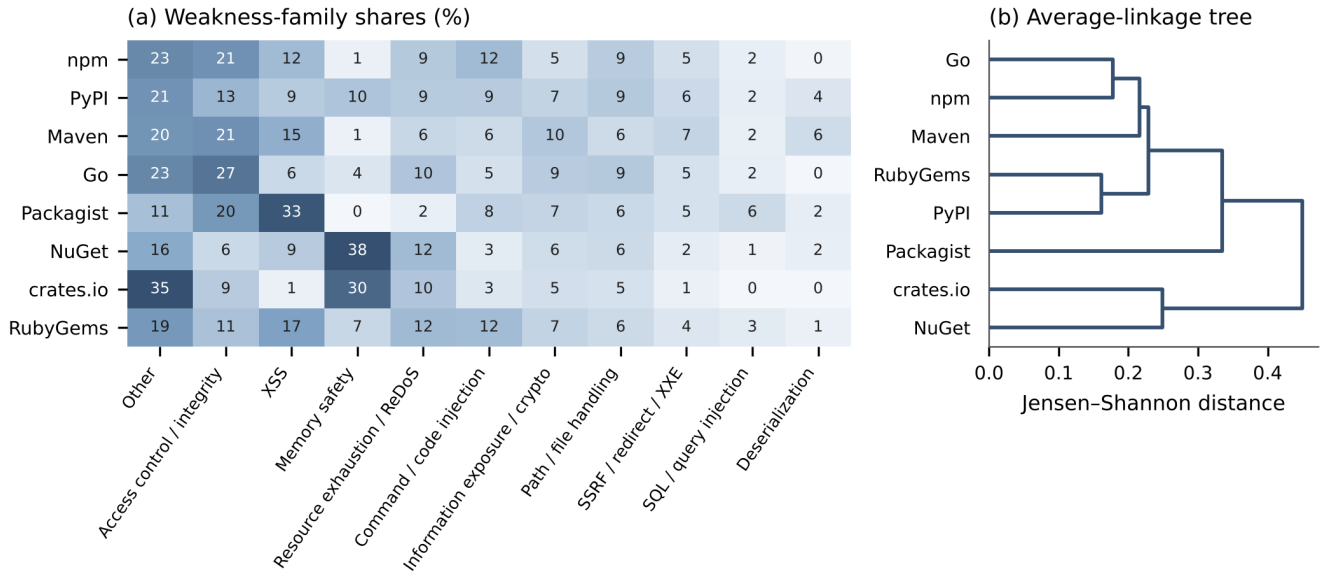


Figure 11. Weakness profiles. (a) Share of each weakness family among classified advisories, by ecosystem. (b) Average-linkage dendrogram on Jensen–Shannon distances.

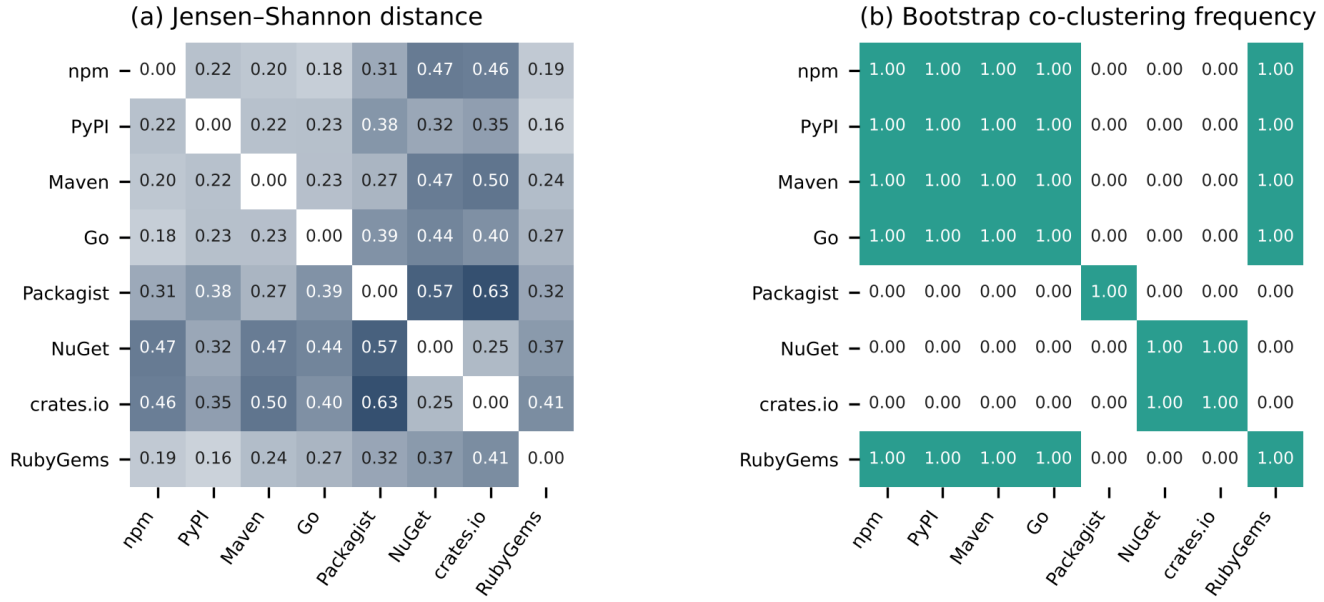


Figure 12. (a) Pairwise Jensen–Shannon distances between ecosystem weakness profiles. (b) Share of 300 bootstrap resamples in which each pair falls in the same of three average-linkage clusters.

5.7 Severity agreement and exploitation (RQ5)

For 9,118 contemporaneous advisories whose CVE has both a GitHub CVSS v3.1 vector and an NVD primary score, NVD scored higher on average by 0.30 points (SD 1.38; Bland–Altman limits −2.40 to 3.00; Figure 13 and Table 9). The two placed the CVE in the same severity band 71.7% of the time, with quadratic-weighted $\kappa_w = 0.67$, a level conventionally read as substantial agreement but one that still leaves more than a quarter of CVEs in different bands. Disagreement leans one way. NVD scored higher in 29.7% of pairs and lower in 16.4%, and 62% of CVEs that GitHub rated Low were rated Medium by NVD; agreement was lower for maintainer advisories (65.0%) than for other advisories (77.9%), whose GitHub score is often copied from or reconciled with NVD's.

Note. Limited to CVE identifiers from 2022 onward, for which NVD records were parsed.

Exploitation evidence is thin. Only 125 reviewed advisories map to CVEs in the KEV catalog, 15 of them with known ransomware use; Maven contributes 51 of the 125, Packagist 23 and PyPI 14, while crates.io, the Rust registry, contributes none at all. The KEV rate rises steeply with GitHub severity, from 0.04% of Moderate advisories to 1.36% of Critical ones, which is consistent with exploitation of open-source packages concentrating on server-side Java and PHP frameworks exposed to the internet.

5.8 IoT-related packages and early warning (RQ6)

IoT packages look different (Table 10, Figure 14, panel a). Critical advisories made up 22.2% of their 135 advisories (95% CI 16.0–29.9) against 13.2% elsewhere, and two-thirds were rated High or Critical against 49.0%. Their affected entries lacked a fixed version more often (18.0% against 11.6%) and more often declared every release up to the fix as affected (64.4% against 50.4%). Maintainer origin was similar. The subset grew faster than the database, from 16 contemporaneous advisories in 2025 to 56 in the first seven months of 2026, with Home Assistant, ESPHome, Node-RED, Apache IoTDB and the OPC UA libraries among the most frequent packages. None of the 135 maps to a CVE in the KEV catalog. With a subset this small the intervals are wide, and a difference of the size observed for Critical advisories would need roughly twice as many IoT advisories to be estimated to within five points.

Note. Percentages with 95% Wilson intervals. IoT-related advisories come mainly from npm (46), PyPI (41) and Maven (39).

The early-warning test asks a narrower question. Of 4,562 episodes published between January and late July 2026, 17.2% were followed by another episode of the same package within seven days, and 47.1% came from packages that had no episode in the training years. Table 11 lists the results. The adaptive score of Algorithm 6 reached an AUC of 0.750 (95% CI 0.732–0.769) and an average precision of 0.420; alerting on its top decile would have caught 31.3% of follow-up disclosures with a precision of 53.7%, three times the base rate. Freezing the background rates at their training values cost 0.06 in AUC (0.688), and the modulated Poisson process with frozen rates did worse again (0.652). Two plain heuristics did as well as the adaptive model: the past episode count reached 0.747, with a paired bootstrap difference from the adaptive score of 0.0001 (−0.005 to 0.006), and the recency of the previous episode reached 0.736.

Note. Training on January 2023 – December 2025 (4,374 packages; branching ratio 0.033, decay 2.60 per day). Base rate 17.2%.

Two points follow. The gain over the frozen model comes from updating each package's background rate, not from the triggering kernel, which is what the small branching ratio of Section 5.3 predicts; at a seven-day horizon the kernel's eight-hour half-life has almost decayed. And the probabilities run low. In the top quintile the score predicted 13.4% and 40.5% of episodes were followed within a week, because the modulation carried forward from December 2025 ($\hat{g} = 1.27$) understates the 2026 level, which reached 3.25 in May. A deployed agent would re-estimate $\hat{g}$ monthly; whether that alone restores calibration can only be tested on data from after July 2026.

6. DISCUSSION

6.1 What the models say together

Table 12 gathers the answers. Read together, they describe a pipeline that changed character in early 2026. Advisories now originate mostly with maintainers, arrive in batches, reach GitHub before NVD and almost always name a fix. The older route has not gone away. In it, GitHub imports an NVD CVE and maps it to a package, and that route still supplies 78.8% of Maven advisories, 80.9% of all unpatched entries and most of the review delay.

6.2 Implications for maintainers, registries and tool builders

For tools that consume the database, the results argue for changing two assumptions. Arrival is bursty at the hour scale. It is stationary at no scale. A scanner that polls daily will see the openclaw-style batch as one event, which is appropriate, but alerting thresholds built on 2025 volumes will be exceeded several times over in 2026 without any change in a given customer's exposure. The modulated Hawkes fit gives a way to tell the difference: the background modulation $\hat{g}(t)$ captures database-wide surges, while $\hat{\alpha}$ captures package-level follow-up, which in these data is small and short-lived. Repeated disclosures for one package within weeks, a pattern that can signal an incomplete fix or a patch bypass, are present (20.8% of episodes) but mostly explained by the package's own background rate.

For registry operators and curators, the remediation results point to a concentrated problem. About 12% of affected entries have no fixed version, but the beta-binomial correlations of 0.26 to 0.75 mean that unpatched entries cluster in packages; a few hundred packages, identifiable from public data with Equation 17, account for most of them. Some of those are mapping artefacts (the Magento case) that a curator can fix in an afternoon. Others are packages without an active maintainer, for which a registry-level deprecation notice would carry more information than another advisory. Critical advisories are the awkward case. Their entries carry higher odds of being unpatched (1.88), and these are the entries scanners flag most loudly while offering the user nothing to

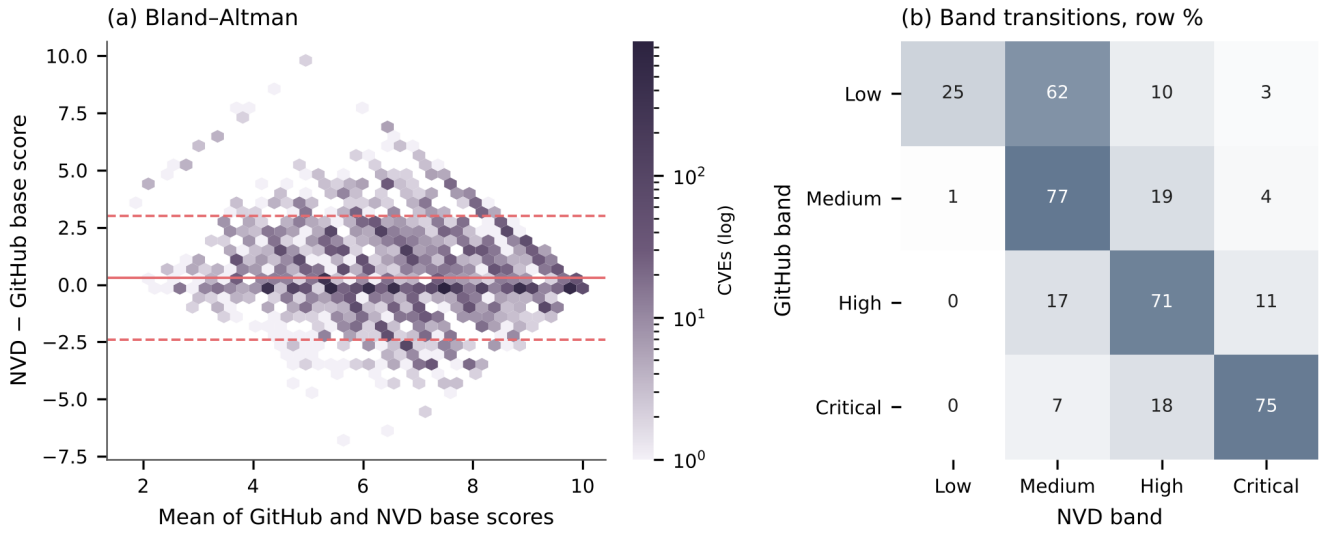


Figure 13. Agreement between GitHub and NVD CVSS v3.1 base scores for the same CVE. (a) Bland–Altman plot with mean difference and 95% limits of agreement, hexagonal bins on a log colour scale. (b) Transition from GitHub severity band (rows) to NVD band (columns), row percentages.

Table 9. GitHub and NVD severity agreement, contemporaneous advisories with both scores

Measure	All	Maintainer advisories	Other advisories
CVEs compared	9,118	4,393	4,725
Mean difference, NVD − GitHub (points)	0.30	0.33	0.27
Same severity band (%)	71.7	65.0	77.9
NVD higher (%)	29.7	—	—
NVD lower (%)	16.4	—	—
Absolute difference ≥ 1 point (%)	34.4	—	—
Quadratic-weighted kappa	0.67	—	—

Table 10. IoT-related packages compared with all other reviewed advisories

Measure	IoT-related	Other
Advisories	135	33,026
Packages	57	13,003
Critical (%)	22.2 (16.0–29.9)	13.2 (12.9–13.6)
High or Critical (%)	66.7 (58.4–74.1)	49.0 (48.5–49.5)
Maintainer advisories (%)	48.9 (40.6–57.2)	43.7 (43.1–44.2)
Unpatched entries (%)	18.0 (13.3–24.1)	11.6 (11.4–11.9)
Entries affected from version 0 (%)	64.4 (57.5–70.8)	50.4 (50.0–50.8)
Advisories with a CVE in KEV	0	125
Contemporaneous advisories, 2025 → 2026 (Jan–Jul)	16 → 56	3,540 → 8,346

upgrade to.

For intelligent monitoring of IoT fleets, the models supply the pieces an automated agent needs and set limits on what it can expect. The background modulation tells the agent when the whole database is surging, the package rates tell it which projects disclose often, and the beta-binomial posteriors tell it where an alert is likely to come without an upgrade path. That last point matters more for IoT than for servers. Packages built for IoT were more often unpatched (18.0%) and more often affected from their first release, which on a device fleet translates into advisories that require a workaround or a replacement component, not a routine over-the-air update. The early-warning result is modest and should be read that way. An adaptive point-process score anticipates follow-up disclosures about as well as counting past episodes, so its value lies in interpretability and calibration, which a heuristic lacks, and in the decomposition into surge, package and follow-up components that an operator can inspect.

For defenders who combine sources, the severity results set expectations. GitHub and NVD agree on the band about seven times in ten, NVD leans higher, and disagreement is greatest for maintainer advisories, which now make up most of the stream and for which the GitHub score is, in effect, the maintainer's own judgement of severity. NIST's April 2026 shift to prioritising a subset of CVEs for enrichment, leaving others not immediately enriched, underscores why GitHub's score can be an important complement when NVD

Table 11. Seven-day early-warning performance on episodes published January–July 2026

Score	AUC (95% CI)	Average precision	Precision, top 10%	Recall, top 10%
Adaptive Hawkes (Algorithm 6)	0.750 (0.732–0.769)	0.420	0.537	0.313
Adaptive Poisson	0.750 (0.729–0.768)	0.415	0.537	0.313
Past episode count	0.747 (0.728–0.765)	0.414	0.535	0.312
Recency of previous episode	0.736 (0.716–0.754)	0.334	0.439	0.255
Modulated Hawkes, frozen background	0.688 (0.666–0.711)	0.333	0.408	0.238
Modulated Poisson, frozen background	0.652 (0.626–0.673)	0.312	0.390	0.227

Table 12. Research questions and principal estimates

Question	Principal estimate	Model
RQ1 Regime change	Break in Feb 2026; monthly mean 280.9 to 1,307.8; regime ratio 4.0 NuGet, 3.8 Go, 0.89 Maven	Binary segmentation; negative binomial
RQ2 Ordering and review	GitHub-first 51.9% in 2026; maintainer OR 88.4; 90.8% of maintainer advisories reviewed at publication	Logistic; quantiles
RQ3 Self-excitation	46.9% of advisories in batches; branching ratio 0.047; half-life 8.1 h	Modulated Hawkes (EM)
RQ4 Concentration and coverage	Tail exponent 2.14; 11.7% unpatched; maintainer OR 0.14; intra-package correlation 0.26–0.75	Power law; logistic; beta-binomial
RQ5 Weaknesses and severity	Three stable clusters; JSD 0.16–0.63; band agreement 71.7%, κw 0.67	Jensen–Shannon; Bland–Altman
RQ6 Early warning and IoT	Adaptive score AUC 0.750 at seven days; IoT Critical 22.2% vs 13.2%, unpatched 18.0% vs 11.6%	Online gamma-updated Hawkes; Wilson intervals

enrichment is missing [10]; our data cannot say whether it is better or worse, only that it differs. KEV overlap is a reminder of scale. Fewer than 0.4% of reviewed advisories concern a CVE known to be exploited, and those cluster in Maven and Packagist server frameworks.

6.3 Threats to validity

Construct validity rests on three operational definitions. "Maintainer advisory" is inferred from a reference URL, which captures repository advisories published through GitHub but would miss a maintainer who reports through another CNA. "Unpatched" means that the advisory names no fixed version, which can be a mapping artefact; the Magento case is one, and we did not attempt to correct such cases package by package. The one-hour episode window is a choice, so we varied it. With windows of 30 minutes, two hours and one day the share of advisories released in batches was 46.1%, 47.6% and 50.7%; even a one-minute window leaves 34.0% in batches.

Internal validity is limited by the snapshot design. Records change. Advisories are edited after publication, and the July 2026 state may differ from what a scanner saw on the day of publication; Git history could recover earlier states and is the obvious extension. The Hawkes model's residual misfit (Section 5.3) means its branching ratio should be read as an order of magnitude. External validity is bounded by the reviewed subset. GitHub reviews only advisories it can map to a supported ecosystem, so unreviewed records, ecosystems outside the eight analysed, and vulnerabilities disclosed only through other databases fall outside these estimates. The 2026 surge is partly driven by a few packages, and whether it reflects lasting change, for instance broader use of automated vulnerability discovery, or a transient burst can only be settled by data from the second half of 2026.

7. CONCLUSION

We modelled the reviewed GitHub Advisory Database as a stochastic process and found a stream that changed regime in February 2026, when monthly volume rose almost fivefold on the strength of maintainer advisories, which now reach GitHub before NVD more often than not, are reviewed at publication and usually name a fix. Batching matters more than triggering. Nearly half of all advisories arrive in same-hour batches, and beyond batching self-excitation within packages

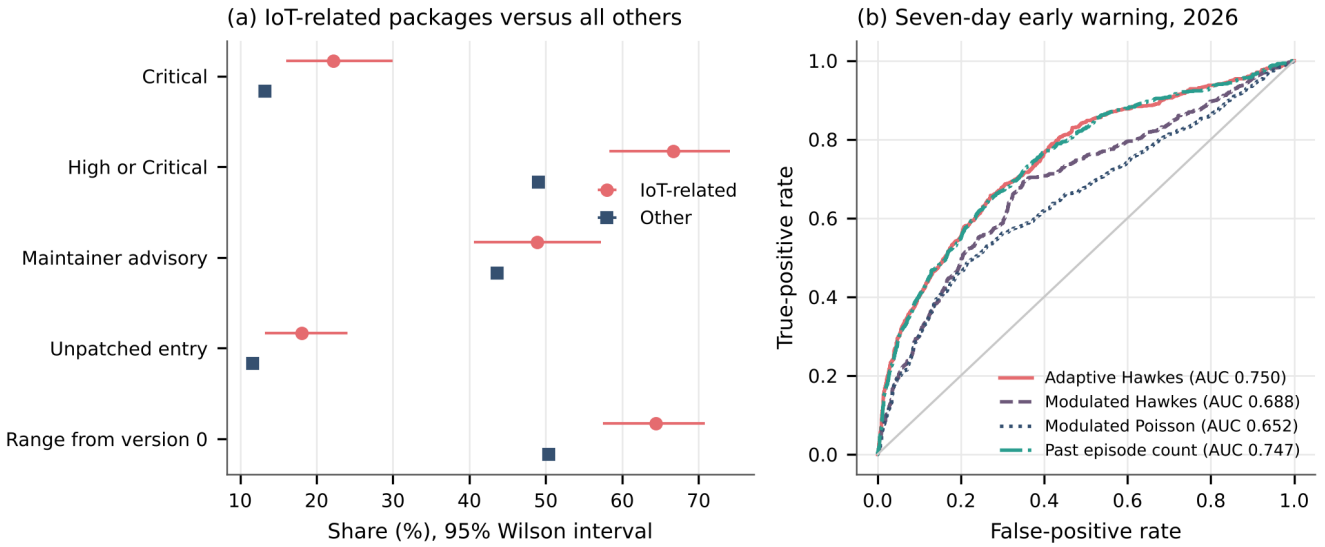


Figure 14. (a) IoT-related packages compared with all other packages on five advisory characteristics, with 95% Wilson intervals. (b) Receiver operating characteristic curves for seven-day early-warning scores on episodes published in 2026.

is weak and short-lived once a shared time-varying background is allowed. Advisories per package are heavy-tailed with an exponent near two, unpatched entries concentrate in a minority of packages, ecosystems fall into three stable groups by weakness profile, and GitHub's severity bands agree with NVD's about seven times in ten. Packages built for IoT carry more Critical and more unpatched advisories than the rest, and an adaptive early-warning score derived from the fitted process ranks follow-up disclosures with an AUC of 0.75, as well as the best heuristic, while running low on probability during the surge. The next data release that matters is the second half of 2026, which will settle whether the new regime persists.

REFERENCES

[1] D. Wermke, N. Wöhler, J. H. Klemmer, M. Fourné, M. Acar, and S. Fahl, "Committed to trust: A qualitative study on security & trust in open source software projects," in *Proceedings of the 2022 IEEE Symposium on Security and Privacy*, pp. 1880–1896, 2022.

[2] M. Antonakakis, T. April, M. Bailey, M. Bernhard, E. Bursztein, J. Cochran, Z. Durumeric, J. A. Halderman, L. Invernizzi, M. Kallitsis, D. Kumar, C. Lever, Z. Ma, J. Mason, D. Menscher, C. Seaman, N. Sullivan, K. Thomas, and Y. Zhou, "Understanding the Mirai botnet," in *Proceedings of the 26th USENIX Security Symposium*, pp. 1093–1110, 2017.

[3] N. Neshenko, E. Bou-Harb, J. Crichigno, G. Kaddoum, and N. Ghani, "Demystifying IoT security: An exhaustive survey on IoT vulnerabilities and a first empirical look on Internet-scale IoT exploitations," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 3, pp. 2702–2733, 2019.

[4] O. Alrawi, C. Lever, M. Antonakakis, and F. Monrose, "SoK: Security evaluation of home-based IoT deployments," in *Proceedings of the 2019 IEEE Symposium on Security and Privacy*, pp. 1362–1380, 2019.

[5] Executive Office of the President, "Improving the nation's cybersecurity (Executive Order 14028)," *Federal Register*, vol. 86, no. 93, pp. 26633–26647, 2021.

[6] N. Imtiaz, S. Thorne, and L. Williams, "A comparative study of vulnerability reporting by software composition analysis tools," in *Proceedings of the 15th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, Art. no. 5, 2021.

[7] Open Source Security Foundation, "OSV schema: Open Source Vulnerability format," version 1.7.0, 2025.

[8] C. Segal, P. Segal, C. E. de Schuller Banjar, F. Paixão, H. S. Borges, P. Silveira Neto, E. S. de Almeida, J. C. S. Santos, A. Kocheturov, G. K. Srivastava, and D. S. Menasché, "Characterizing and modeling the GitHub Security Advisories review pipeline," in *Proceedings of the 23rd ACM/IEEE International Conference on Mining Software Repositories*, pp. 311–322, 2026.

[9] U.S. Department of Commerce, Office of Inspector General, "Evaluation of NIST's Management of the National Vulnerability Database," Report No. OIG-26-020-I, May 26, 2026.

[10] National Institute of Standards and Technology, "NIST Updates NVD Operations to Address Record CVE Growth," Apr. 15, 2026.

[11] M. Ohm, H. Plate, A. Sykosch, and M. Meier, "Backstabber's knife collection: A review of open source software supply chain attacks," in *Detection of Intrusions and Malware, and Vulnerability Assessment (DIMVA 2020)*, Springer, pp. 23–43, 2020.

[12] P. Ladisa, H. Plate, M. Martinez, and O. Barais, "SoK: Taxonomy of attacks on open-source software supply chains," in *Proceedings of the 2023 IEEE Symposium on Security and Privacy*, pp. 1509–1526, 2023.

[13] A. Decan, T. Mens, and E. Constantinou, "On the impact of security vulnerabilities in the npm package dependency network," in *Proceedings of the 15th International Conference on Mining Software Repositories*, pp. 181–191, 2018.

[14] M. Alfadel, D. E. Costa, and E. Shihab, "Empirical analysis of security vulnerabilities in Python packages," *Empirical Software Engineering*, vol. 28, no. 3, Art. no. 59, 2023.

[15] G. A. A. Prana, A. Sharma, L. K. Shar, D. Foo, A. E. Santosa, A. Sharma, and D. Lo, "Out of sight, out of mind? How vul-

nerable dependencies affect open-source projects," *Empirical Software Engineering*, vol. 26, no. 4, Art. no. 59, 2021.

[16] N. Imtiaz, A. Khanom, and L. Williams, "Open or sneaky? Fast or slow? Light or heavy?: Investigating security releases of open source packages," *IEEE Transactions on Software Engineering*, vol. 49, no. 4, pp. 1540–1560, 2023.

[17] M. Zimmermann, C.-A. Staicu, C. Tenny, and M. Pradel, "Small world with high risks: A study of security threats in the npm ecosystem," in *Proceedings of the 28th USENIX Security Symposium*, USENIX Association, pp. 995–1010, 2019.

[18] C. Yang-Smith and A. Abdellatif, "Tracing vulnerabilities in Maven: A study of CVE lifecycles and dependency networks," arXiv:2502.04621, 2025.

[19] R. G. Kula, D. M. German, A. Ouni, T. Ishio, and K. Inoue, "Do developers update their library dependencies? An empirical study on the impact of security advisories on library migration," *Empirical Software Engineering*, vol. 23, no. 1, pp. 384–417, 2018.

[20] I. Pashchenko, H. Plate, S. E. Ponta, A. Sabetta, and F. Massacci, "Vulnerable open source dependencies: Counting those that matter," in *Proceedings of the 12th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, Art. no. 42, 2018.

[21] A. G. Hawkes, "Spectra of some self-exciting and mutually exciting point processes," *Biometrika*, vol. 58, no. 1, pp. 83–90, 1971.

[22] T. Ozaki, "Maximum likelihood estimation of Hawkes' self-exciting point processes," *Annals of the Institute of Statistical Mathematics*, vol. 31, no. 1, pp. 145–155, 1979.

[23] Y. Ogata, "Statistical models for earthquake occurrences and residual analysis for point processes," *Journal of the American Statistical Association*, vol. 83, no. 401, pp. 9–27, 1988.

[24] A. Veen and F. P. Schoenberg, "Estimation of space–time branching process models in seismology using an EM-type algorithm," *Journal of the American Statistical Association*, vol. 103, no. 482, pp. 614–624, 2008.

[25] E. N. Brown, R. Barbieri, V. Ventura, R. E. Kass, and L. M. Frank, "The time-rescaling theorem and its application to neural spike train data analysis," *Neural Computation*, vol. 14, no. 2, pp. 325–346, 2002.

[26] A. Clauset, C. R. Shalizi, and M. E. J. Newman, "Power-law distributions in empirical data," *SIAM Review*, vol. 51, no. 4, pp. 661–703, 2009.

[27] Q. H. Vuong, "Likelihood ratio tests for model selection and non-nested hypotheses," *Econometrica*, vol. 57, no. 2, pp. 307–333, 1989.

[28] J. Alstott, E. Bullmore, and D. Plenz, "powerlaw: A Python package for analysis of heavy-tailed distributions," *PLOS ONE*, vol. 9, no. 1, Art. no. e85777, 2014.

[29] D. A. Williams, "The analysis of binary responses from toxicological experiments involving reproduction and teratogenicity," *Biometrics*, vol. 31, no. 4, pp. 949–952, 1975.

[30] B. Efron and C. Morris, "Data analysis using Stein's estimator and its generalizations," *Journal of the American Statistical Association*, vol. 70, no. 350, pp. 311–319, 1975.

[31] J. Lin, "Divergence measures based on the Shannon entropy," *IEEE Transactions on Information Theory*, vol. 37, no. 1, pp. 145–151, 1991.

[32] D. M. Endres and J. E. Schindelin, "A new metric for probability distributions," *IEEE Transactions on Information Theory*, vol. 49, no. 7, pp. 1858–1860, 2003.

[33] R. R. Sokal and C. D. Michener, "A statistical method for evaluating systematic relationships," *University of Kansas Science Bulletin*, vol. 38, pp. 1409–1438, 1958.

[34] A. J. Scott and M. Knott, "A cluster analysis method for grouping means in the analysis of variance," *Biometrics*, vol. 30, no. 3, pp. 507–512, 1974.

[35] R. Killick, P. Fearnhead, and I. A. Eckley, "Optimal detection of changepoints with a linear computational cost," *Journal of the American Statistical Association*, vol. 107, no. 500, pp. 1590–1598, 2012.

[36] A. C. Cameron and P. K. Trivedi, *Regression Analysis of Count Data*, 2nd ed. Cambridge, U.K.: Cambridge University Press, 2013.

[37] J. Cohen, "Weighted kappa: Nominal scale agreement provision for scaled disagreement or partial credit," *Psychological Bulletin*, vol. 70, no. 4, pp. 213–220, 1968.

[38] Forum of Incident Response and Security Teams, "Common Vulnerability Scoring System version 3.1: Specification document," 2019.

[39] Forum of Incident Response and Security Teams, "Common Vulnerability Scoring System version 4.0: Specification document," 2023.

[40] Cybersecurity and Infrastructure Security Agency, "Binding Operational Directive 22-01: Reducing the significant risk of known exploited vulnerabilities," U.S. Department of Homeland Security, 2021.