A Multi-Scanner Census of the Linux Operating-System Base Images of Docker Hub

Cristhian Kapelinski[1], **Diego Kreutz**[1]

[1]AI Horizon Labs, Federal University of Pampa (UNIPAMPA)
{cristhianavila.aluno,diegokreutz}@unipampa.edu.br

Abstract. *Most container images build on an operating-system (OS) base image, yet the security posture of these bases is usually characterized with a single scanner. We present a recent multi-scanner census of Docker Hub's Linux OS base images:* ***5,606*** *unique* `amd64` *images across* ***20*** *distributions, scanned with* ***14*** *open-source tools. The vulnerability count per image varies by an order of magnitude across distributions and scales with image age more than with package count (standardized* $\beta \approx 0.44$ *vs.* 0.15*), while image size points the other way and popularity explains almost nothing; about one in twelve historically published images can no longer be pulled by a current Docker Engine. The four package-vulnerability engines show pairwise agreement of at most* ***0.36*** *(Jaccard), and raw secret and malware detection rates above* ***80%*** *do not survive manual validation: none of* ***1,100*** *sampled detections per axis was a true positive.*

1. Introduction

Containers are now a default unit of software delivery: Docker puts its developer community above 20 million[1], and its 2025 survey reports container use by 92% of IT professionals surveyed[2]. Underneath them sits Docker Hub, the largest public registry, which held 8.3 million image repositories and 318 billion cumulative pulls already in 2021[3] and has grown since. Most of those images begin with a single line, `FROM` an operating-system (OS) base image, and a few bases carry most of that volume: `alpine`, `ubuntu`, `debian` and `busybox` each record over a billion pulls, and Debian, Alpine and Ubuntu account for most base-image usage[4]. A flaw in a base image is inherited, unchanged, by every image built on it [Shu et al. 2017], so the posture of these few bases reaches far beyond them.

This inherited exposure is consequential. Base images sit at the root of the software supply chain, so their known vulnerabilities are a supply-chain risk in their own right, and the scale is documented: popular Docker Hub images carry hundreds of known vulnerabilities on average, many of them years old[5], against a backdrop of record CVE disclosure (48,185 CVEs published in 2025)[6]. In addition, some of the most-pulled OS bases are no longer maintained: every official `centos` tag reached end-of-life (EOL) in June 2024, yet the repository is still the fifth most-pulled of our corpus, at 1.2 billion pulls (Section 4.4).

Despite this central role, the security posture of OS base images has usually been characterized with a *single* scanner and on samples that age quickly: the large-scale

[1] Docker, State of App Dev, 2024 [2] Docker, App Dev survey, 2025 [3] Docker Index, 2021 [4] Anchore, OS breakdown of Docker Hub, 2024 [5] NetRise, Supply chain visibility study, 2024 [6] J. Gamblin, 2025 CVE data review, 2026

Docker Hub measurements have typically used one detector [Shu et al. 2017, Zerouali et al. 2019, Liu et al. 2020, Shi et al. 2025], and the studies that *do* compare scanners have done so on focused samples [Mills et al. 2023, Boles et al. 2024]. No recent work measures the Linux OS base images specifically, with a heterogeneous multi-scanner battery, as a current census.

Contributions. We present (i) a recent multi-scanner census of Docker Hub's Linux OS base images (Section 3); (ii) a per-distribution posture and a result that contradicts a common assumption: vulnerability load tracks image *age* far more than package count ($\beta \approx 0.44$ vs. 0.15), so a smaller base is not a safer one (Sections 4.1, 4.2, 4.5); (iii) an image-longevity finding (Section 4.4); (iv) as a secondary result, a four-engine divergence measurement on the OS-package layer (Section 4.3); and (v) a public multi-scanner dataset, to our knowledge the first focused on OS base images: the per-image findings of all 14 scanners over the census, released with the full pipeline[7] so the corpus can be re-analyzed under other metrics or scanners without repeating the costly scan.

2. Related Work

Table 1 positions our work along the six axes that bound what a Docker Hub measurement can conclude: corpus size, tool count, dimensions, source, OS-base targeting, and cross-scanner comparison.

Table 1. Prior OS/base-image and container-scanner measurements.

Study	Images	Tools	Dims	Source	OS	X-sc.
[Shu et al. 2017]	356,218	1	V	Hub crawl	✗	✗
[Liu et al. 2020]	2.2 M*	1	V,M	Hub crawl	✗	✗
[Zerouali et al. 2021]	140,498	1‡	V	Hub, Debian	✓	✗
[Haque et al. 2022]	261	1	V	Hub, official	✓	✗
[Dahlmanns et al. 2023]	337,171	1	S	Hub crawl	✗	✗
[Mills et al. 2023]	380	6	V	Hub categories	✗	✓
[Boles et al. 2024]	927	2	V	Hub official	✗	✓
[Bufalino et al. 2025]	20†	7	V,B	Hub Deb/Alp	✓	✓
[Shi et al. 2025]	33,952*	1	V,S,C,M	Hub pull+dep.	✗	✗
[Kapelinski et al. 2026]	52,895	6	V,B,S,C	Hub exposure	✗	✓
[Kapelinski and Kreutz 2026b]	2,879	6	V,B,S,C	Hub uniform	✗	✓
This work	**5,606**	**14**	**V,B,S,C,M**	**Hub category**	✓	✓

Dims: V vuln, B SBOM (software bill of materials), S secrets, C config, M malware. X-sc.: cross-scanner. ∗ vulnerability scan on a subset; † top-20 Debian/Alpine image families; ‡ Debian Security Tracker (metadata), not a scanner.

A rich line of work has charted the security of Docker Hub. The first large-scale study scanned 356,218 images and showed that vulnerabilities propagate from a base image to everything built on it [Shu et al. 2017]; the scale later reached 2.2 million images [Liu et al. 2020]. *Technical lag*, how far behind the latest package versions an image is, was applied to 7,380 Debian-based images [Zerouali et al. 2019] and generalized to a multi-dimensional analysis over 140,000 [Zerouali et al. 2021]. Later work analyzed 2,500 images across image classes [Wist et al. 2021], examined how images for the same system differ across base distributions [Ibrahim et al. 2020], and measured influential images selected by both pull count and dependency weight over a layer-derived dependency graph [Shi et al. 2025]. Most relevant to us, a study of 261 official base images characterized the exploitability and impact of their vulnerabilities [Haque et al. 2022] and reported a result we revisit in Section 4.5: highly exploitable vulnerabilities are *more*

[7] github.com/ChimangoScan/os-census; the per-image dataset is linked from the repository.

frequent in minimal images such as Alpine, while larger distributions carry the higher-impact ones. These studies established the prevalence and propagation of vulnerabilities in the ecosystem; each draws on a single vulnerability detector, and our multi-scanner design complements them by also quantifying how much the measured posture depends on the chosen scanner. Prior scanner comparisons on focused samples report large disagreement [Boles et al. 2024, Bufalino et al. 2025], which we measure at OS-base census scale (Section 4.3). Our recent studies scanned the highest-exposure Docker Hub images and a uniform random sample of the registry with six scanners [Kapelinski et al. 2026, Kapelinski and Kreutz 2026b]; both report vulnerability findings concentrated in a single engine and secret detections that hand-labeling refutes, which we re-examine here on the OS bases with a broader battery. We are not aware of a comparable census, nor of one that quantifies the prevalence and the pull volume of *end-of-life* OS images.

3. Methodology

We take as our corpus the operating-system base images Docker Hub lists under its *Operating systems* category, crawled in 2026: **20** distributions with an `amd64` image (`alpine`, `ubuntu`, `debian`, `centos`, `busybox`, `amazonlinux`, `fedora`, `oraclelinux`, `rockylinux`, `almalinux`, `photon`, `archlinux`, `tumbleweed`, `leap`, `kali-rolling`, `mageia`, `alt`, `cirros`, `clearlinux`, `sl`); `rockylinux` is published under two namespaces, so the corpus spans 21 repositories. Deduplicating tags by `amd64` content digest (e.g. `24.04`, `noble` and `latest` are one image) leaves **5,606** unique images, the unit of measurement. An OS base image is a static root filesystem with no application source code and no running service, so we select the static-analysis axes that apply to such an artifact and run *14* open-source scanners.[8] *Software bill of materials (SBOM)*: Syft and cdxgen. *Package vulnerabilities (software composition analysis, SCA)*: Trivy, Grype, OSV-Scanner and Clair. *Image hardening*: Dockle and Checkov. *Secrets*: TruffleHog, Gitleaks, detect-secrets and Whispers. *Malware and indicators of compromise (IOC)*: ClamAV and YARAHunter. We exclude the static application security testing (SAST), dynamic application security testing (DAST) and network axes, which need application source or a running service a base image lacks.

Pipeline. All scans run through the same open-source orchestration engine as our measurement of high-exposure Docker Hub images [Kapelinski et al. 2026], and the engine is released with the artifact. It resolves each tag to its `amd64` content digest, pulls the image pinned by that digest, and exports its filesystem once to a tar archive that all 14 scanners read, so result differences reflect the scanner rather than the pull. Raw scanner outputs are kept verbatim, and a scan queue distributes the images across worker machines.

Counting. Per-image vulnerability counts sum the findings of the four package-vulnerability engines, preserving each engine's own severity assignment. Because the engines overlap, this sum counts a CVE once per engine that reports it: across the census the four engines report 1,025,210 (image, CVE) pairs, which reduce to 710,739 when each CVE is counted once per image (a $1.44\times$ inflation). The per-image ranking is nearly unchanged under this deduplication (Spearman rank correlation $\rho = 0.86$ between summed and deduplicated counts, where 1 is an identical ranking), and the cross-engine analysis of Section 4.3 operates on deduplicated (image, CVE) pairs.

[8] The released configuration records each image reference and invocation. Several references use floating tags, so the released outputs, rather than the tags, are the immutable record of the campaign.

4. Results

4.1. RQ1: Security posture by distribution

Critical- and high-severity vulnerability findings vary by an order of magnitude across distributions (Figure 1(a)). The severity mix also changes across distributions (Fig-

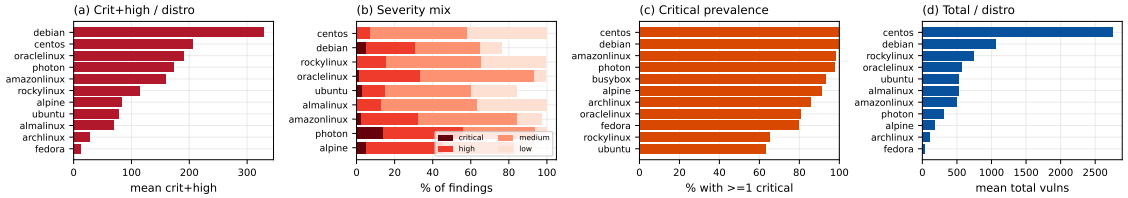

Figure 1. Findings by distribution: (a) mean critical+high per image; (b) severity mix (% of findings); (c) share of images with a critical finding; (d) mean total per image.

ure 1(b)), as do the share of images with a critical finding (Figure 1(c)) and the mean total count (Figure 1(d)). Debian carries the most (mean ≈329 critical+high per image, n=2,641), followed by CentOS (≈206), OracleLinux (≈190), Photon (≈174), AmazonLinux (≈160) and RockyLinux (≈128); Alpine, Ubuntu and AlmaLinux sit lower (≈84, 79 and 69), and Fedora, Arch, ALT, Kali, Mageia and openSUSE Tumbleweed report few or none. Within-distribution variance is large (Debian's standard deviation (SD), ≈375, exceeds its mean) and some repositories are small (CentOS $n = 10$), so these are tendencies, not tight estimates.

4.2. RQ2: Staleness vs. vulnerability

Vulnerability count tends to rise with image age (Figure 2(a)). Images rebuilt within the

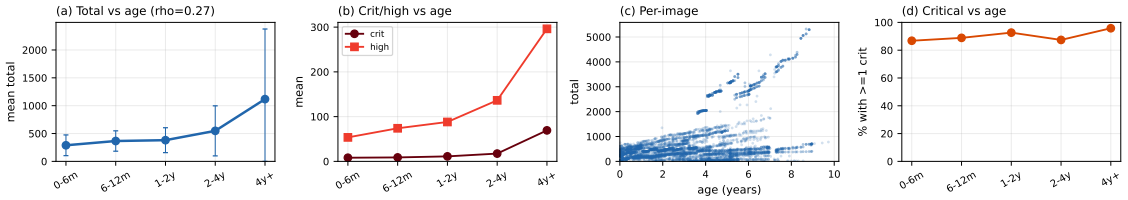

Figure 2. Image age vs. (a) mean total findings; (b) mean critical and high findings; (c) per-image totals; (d) share of images with a critical finding.

last six months carry a mean of ≈289 total vulnerability findings, climbing through ≈366 (six to twelve months), ≈381 (one to two years) and ≈548 (two to four years) to ≈1,118 after four years. Critical and high findings rise with the same ordering (Figure 2(b)), while the per-image scatter remains wide (Figure 2(c)); their rank correlation is positive but modest (Spearman $\rho = 0.27$, $p < 0.001$). The share of images with a critical finding is higher for the oldest images than for the newest (95.8% vs. 86.8%), though it does not rise monotonically (Figure 2(d)). Thus, age tracks higher scanner counts across 20 distributions, but does not determine an individual image's count, consistent with the technical-lag framing of [Zerouali et al. 2019].

4.3. RQ3: Inter-scanner divergence on OS packages

The four package-vulnerability engines show low pairwise agreement (Figure 3), measured as the Jaccard index: the share of two engines' combined findings that both report,

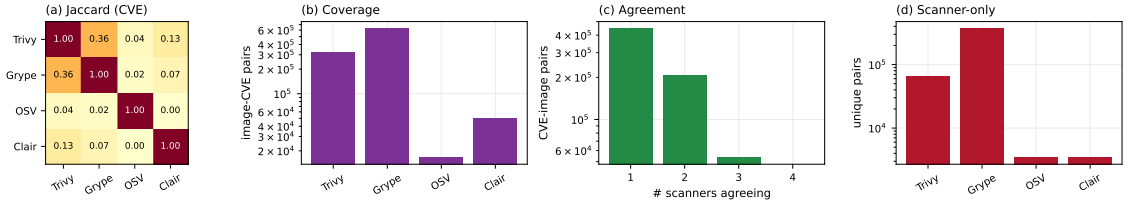

Figure 3. Cross-engine divergence over (image, CVE) pairs: (a) pairwise Jaccard; (b) pairs reported per engine; (c) how many engines report each pair; (d) pairs reported by one engine only.

1 identical and 0 disjoint. We compare (image, CVE) pairs because Clair names source packages while Trivy and Grype name binary packages. Trivy and Grype agree most, at **0.36**; the remaining pairs range from zero to 0.13 (Figure 3(a)). Coverage differs accordingly (Figure 3(b)): Grype and Trivy report 636k and 323k pairs, OSV-Scanner reports 16k language-ecosystem pairs, and Clair reports 50k pairs from its supported distribution feeds. Most pairs appear in only one engine (Figure 3(c)), with Grype contributing the largest scanner-only set (Figure 3(d)). The reported posture therefore depends strongly on the chosen scanner.

4.4. RQ4: End-of-life and un-pullable images

The census shows two image-longevity findings. About **one in twelve** (8%) of the 5,606 digest-pinned images fail to pull on a current Docker Engine because they use the retired manifest schema 1. The rate exceeds half for `centos` and `busybox`, reaches about a third for `ubuntu`, and stays below 3% for `debian`, `archlinux`, and `rockylinux` (Figure 4(a)); the absolute counts show where this loss is concentrated (Figure 4(c)). Meanwhile, repository pull counts remain high (Figure 4(b)): every `centos` tag has been EOL since June 2024, yet the repository records over a billion pulls. The joint view (Figure 4(d)) shows that popularity does not protect historical images from becoming un-pullable.

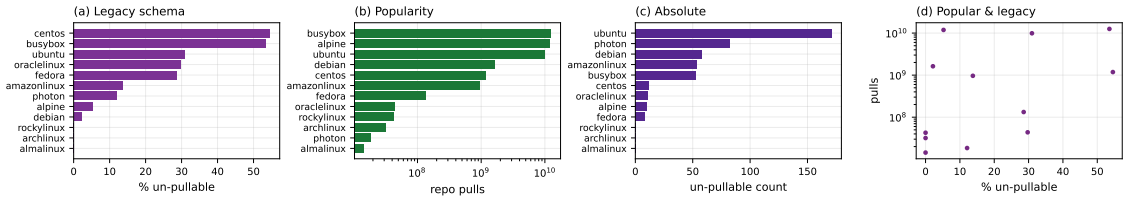

Figure 4. Legacy-manifest images: (a) % un-pullable per repository; (b) repository pulls (log scale); (c) absolute un-pullable count; (d) % un-pullable vs. pulls.

4.5. RQ5: What drives the vulnerability load?

The common assumption that a smaller image is a safer one is not supported by the data (Figure 5(a)). The relationship between package count and vulnerability count is not monotonic: BusyBox, with a single package, carries only a handful of findings, but Mageia, with around 250 packages, carries about a dozen, while Debian carries hundreds of critical and high findings with fewer packages than Mageia. This refines [Haque et al. 2022]: on our multi-scanner data the vulnerability count is governed more by a distribution's advisory coverage and patch cadence than by image size. A multiple regression makes the ranking explicit: with age and package count standardized, image age

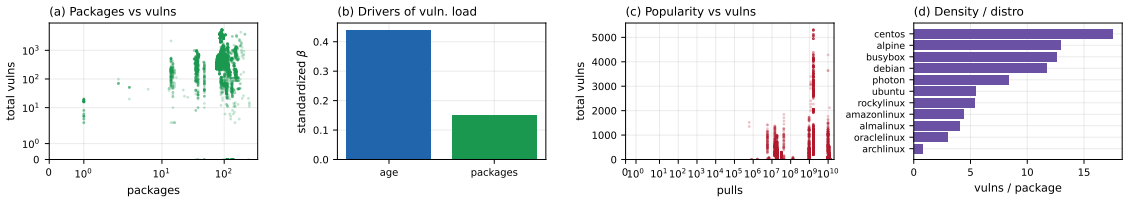

Figure 5. Drivers of the vulnerability load: (a) packages vs. total findings; (b) standardized coefficients for age and package count; (c) pulls vs. total findings; (d) findings per package by distribution.

dominates ($\beta \approx 0.44$) over package count ($\beta \approx 0.15$), the two coefficients plotted in Figure 5(b); adding image size as a third predictor yields a negative coefficient ($\beta \approx -0.23$), so a larger image does not mean more findings once age and packages are controlled. Within individual distributions the drivers vary: in some of them package count overtakes age ($\beta \approx 0.53$ for Ubuntu, 0.65 for AlmaLinux), so the per-distribution variation dominates. Popularity adds no explanatory power: pull count contributes almost nothing once age and size are controlled ($\beta_{\text{pulls}} \approx -0.07$), and the per-image scatter shows no trend against pulls (Figure 5(c)), so the most-pulled bases carry vulnerability counts similar to the rest. Findings per package vary sharply by distribution (Figure 5(d)).

4.6. Beyond vulnerabilities: secrets, misconfiguration, malware

The battery also covers three non-vulnerability axes. *Misconfiguration:* Dockle and Checkov flag a Center for Internet Security (CIS)-style issue in essentially every image, mainly a missing `HEALTHCHECK`, disabled content trust, or execution as `root`; these are expected properties of a base image. *Secrets and malware:* TruffleHog/Gitleaks flag **83%** of images and YARA-Hunter **85%**, but manual validation of **1,100** detections per axis found no real secret or malware (95% upper bound $\leq 0.35\%$; Appendix A). Hits were test keys, checksums, examples, or benign binaries, below the validated rates of [Dahlmanns et al. 2023, Shi et al. 2025]. Raw alerts therefore require triage before operational use.

5. Discussion and conclusion

Implications. Base selection should favor recent, maintained images over merely small or popular ones. Registries can expose rebuild and EOL status; organizations can allowlist bases, combine scanners, and use reachability analysis [Zhou et al. 2026] to filter alerts by runtime relevance. Future censuses should track patch latency, weight downstream exposure [Kapelinski et al. 2026], cover omitted axes such as cryptographic posture [Kapelinski and Kreutz 2026a], and include `linux/arm64`. **Limitations.** Small-repository estimates are tendencies; the corpus covers only Docker Hub's `linux/amd64` OS category, and package findings lack manual adjudication. Advisory-feed differences may affect cross-distribution counts, and floating scanner tags and refreshed databases prevent exact replay. **Conclusion.** More packages do not imply more reported vulnerabilities: age dominates package count, image size points the other way, and popularity explains little. The multi-scanner design separates image factors from tool effects, and the low engine agreement it exposes means one scanner's count does not measure posture.[9]

[9] This work was partially supported by Capes, CNPq (Grant No. 409743/2025-9), and FAPERGS (Grant Nos. 24/2551-0001368-7 and 24/2551-0000726-1). We thank the anonymous reviewers of SBSeg 2026 for their suggestions and comments. The authors also acknowledge the use of Generative AI as a supporting tool for reviewing and improving the code and manuscript.

References

Boles, B. et al. (2024). Deciphering discrepancies: A comparative analysis of Docker image security. In *SCAM*. IEEE.

Bufalino, J. et al. (2025). SBOMproof: Beyond alleged SBOM compliance for supply chain security of container images. arXiv:2510.05798.

Dahlmanns, M. et al. (2023). Secrets revealed in container images: An internet-wide study on occurrence and impact. In *ASIA CCS '23*, pages 797–811. ACM.

Haque, M. U. et al. (2022). Well begun is half done: An empirical study of exploitability and impact of base-image vulnerabilities. In *SANER*, pages 1066–1077. IEEE.

Ibrahim, M. H. et al. (2020). Too many images on DockerHub! how different are images for the same system? *Empirical Softw. Eng.*, 25(5):4250–4281.

Kapelinski, C. and Kreutz, D. (2026a). CryptoCensus: Cryptographic posture and post-quantum readiness of Docker Hub. In *WTICG/SBSeg 2026*. SBC.

Kapelinski, C. and Kreutz, D. (2026b). A uniform random-sample security measurement of Docker Hub images. In *SBSeg 2026*. SBC.

Kapelinski, C., Machado, B., and Kreutz, D. (2026). Vulnerabilities, secrets and misconfiguration in the highest-exposure Docker Hub images. *arXiv preprint arXiv:2608.02669*.

Liu, P. et al. (2020). Understanding the security risks of Docker Hub. In *Computer Security – ESORICS 2020*, pages 257–276. Springer.

Mills, A. et al. (2023). Longitudinal risk-based security assessment of Docker software container images. *Comput. Secur.*, 135:103478.

Shi, H. et al. (2025). Dr. Docker: A large-scale security measurement of Docker image ecosystem. In *WWW '25*. ACM.

Shu, R. et al. (2017). A study of security vulnerabilities on Docker Hub. In *CODASPY '17*, pages 269–280. ACM.

Wist, K. et al. (2021). Vulnerability analysis of 2500 Docker Hub images. In *Advances in Security, Networks, and Internet of Things*, pages 307–327. Springer.

Zerouali, A. et al. (2019). On the relation between outdated Docker containers, severity vulnerabilities, and bugs. In *SANER 2019*, pages 491–501. IEEE.

Zerouali, A. et al. (2021). A multi-dimensional analysis of technical lag in Debian-based Docker images. *Empirical Softw. Eng.*, 26(2):19.

Zhou, L., Dacier, M., and Konstantinou, C. (2026). A reality check on SBOM-based vulnerability management: An empirical study and a path forward. In *CODASPY '26*. ACM.

A. Manual validation protocols

We drew seed-fixed, stratified samples from the running-census snapshot and manually labeled every item; the artifact contains the samples, verdicts, and selection scripts. All 1,100 of 26,892 sampled TruffleHog/Gitleaks detections and all 1,100 of 13,348 sampled YARA-Hunter matches were false positives (Wilson 95% true-positive upper bound $\leq 0.35\%$ for each population). The final census contains 44,339 secret and 20,157 YARA detections.