Container image hardening: security gain or measurement illusion?

How I will judge whether fewer reported vulnerabilities really means a safer image

Valid Al-Dzhazzar · BJ26VH · PROM05 A1 · blog.bywaleed.com

EVALUATION · TOOLS AND TECHNIQUES

1 The question

Does making a container image smaller really make it safer, or does it only hide software from the scanners?

Scanner sees what it reports ≠? **Image holds** what is really inside

Assumed attacker: can already run code inside the container. A severity score (CVSS) signals risk; it does not prove it.

Haque and Babar (2022); Javed and Toor (2021)

2 The experiment

3 apps Python · Node.js · Go × **5 build styles** Full · Slim · Alpine · Multi-stage · Distroless = **15 images**

Trivy scanner 1 · **Grype** scanner 2 · **Inventory** files on disk · **Compare** app by app

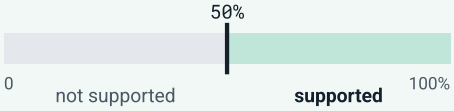
Quantitative experiment: Leavy (2017); Churakova, Ekstedt and Schmid (2026)

3 How each hypothesis is judged

RULES FIXED BEFORE ANY DATA

H1 · Fewer vulnerabilities

Smaller styles vs Slim: 50%+ fewer high and critical flaws, on both scanners, in 2 of 3 apps?



H2a · Hidden software

Multi-stage, Distroless: do flaws both scanners miss equal 10%+ of the drop, in 2 of 3 apps?



H2b · Fixable only

Counting only flaws with a fix, does 25%+ of the drop disappear, in 2 of 3 apps?



H1 is supported or not supported. With no drop, hidden software is still counted. Two sensitivity checks sit beside each verdict. Nosek et al. (2018)

4 What I measure, and with what

- High and critical vulnerabilities, layer by layer
- Attack surface: programs, libraries, shell, setuid files
- Build time (median of 10 builds) and image size
- Failures, and the time to find a planted fault

Cost: vs Slim; over 20% slower to build counts against it.

`Docker · Trivy · Grype · Syft · cve-bin-tool · Python`

Georges, Buytaert and Eeckhout (2007)

5 Can I trust the result?

Right thing measured? Inventory is the reference; unknown files flagged

Fair comparison? Frozen data; every scan run twice

How far it reaches? Shows it can happen, not how often

Honest numbers? Described per app; no significance claims

Wohlin et al. (2024); Ralph (2021)

6 Testing my advisor tool

5 unseen Dockerfiles → **Advisor** builds and scans → **Pick** or decline

Rule: fewest serious vulnerabilities, ≤20% slower; declines if scanners disagree.
Judged on: picks that run, warning accuracy (precision, recall), overturned picks, run time.
Targets (new): all picks run, none overturned, beats "always Slim" in 3 of 5, <30 min.

Henkel et al. (2020)

7 Managing the project



- GitHub Projects: a Kanban board by phase, plus a timeline
- Risk register: likelihood × impact, with early warning signs
- Integrity log · no human participants · 90-day disclosure

Wingate (2025); Phelps, Fisher and Ellis (2007); Cox (2008); BCS (2022)

Evidence scans, inventory, timings → **Verdicts** each hypothesis, per app → **Worth using? (operationally hardened)** Meets H1, hidden software <10%, fixable-only loss <25%, runs, ≤20% slower → **Guidance** which build style, at what cost

1. BCS, The Chartered Institute for IT (2022) BCS Code of Conduct. Version 8. Swindon: BCS.
2. Churakova, Y., Ekstedt, M. and Schmid, L. (2026) 'Vexed by VEX tools: consistency evaluation of container vulnerability scanners', in Al-Mallah, R. et al. (eds) Foundations and Practice of Security: FPS 2025. Lecture Notes in Computer Science, 16402. Cham: Springer, pp. 139-156.
3. Cox, L.A. (2008) 'What's wrong with risk matrices?', Risk Analysis, 28(2), pp. 497-512.
4. Georges, A., Buytaert, D. and Eeckhout, L. (2007) 'Statistically rigorous Java performance evaluation', Proceedings of the 22nd ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages and Applications (OOPSLA '07). New York: ACM, pp. 57-76.
5. Haque, M.U. and Babar, M.A. (2022) 'Well begun is half done: an empirical study of exploitability & impact of base-image vulnerabilities', 2022 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER). IEEE, pp. 1066-1077.
6. Henkel, J., Bird, C., Lahiri, S.K. and Reps, T. (2020) 'Learning from, understanding, and supporting DevOps artifacts for Docker', Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering (ICSE '20). New York: ACM, pp. 38-49.
7. Javed, O. and Toor, S. (2021) 'Understanding the quality of container security vulnerability detection tools', arXiv preprint arXiv:2101.03844.
8. Leavy, P. (2017) Research design: quantitative, qualitative, mixed methods, arts-based, and community-based participatory research approaches. New York: The Guilford Press.
9. Nosek, B.A., Ebersole, C.R., DeHaven, A.C. and Mellor, D.T. (2018) 'The preregistration revolution', Proceedings of the National Academy of Sciences, 115(11), pp. 2600-2606.
10. Phelps, R., Fisher, K. and Ellis, A. (2007) Organizing and managing your research: a practical guide for postgraduates. London: SAGE.
11. Ralph, P. (ed.) (2021) ACM SIGSOFT empirical standards for software engineering research. Version 0.2.0. arXiv:2010.03525v2.
12. Wingate, L.M. (2025) Project management for research and development: guiding innovation for positive R&D outcomes. 2nd edn. Boca Raton, FL: CRC Press.
13. Wohlin, C., Runeson, P., Höst, M., Ohlsson, M.C., Regnell, B. and Wesslén, A. (2024) Experimentation in software engineering. 2nd edn. Berlin, Heidelberg: Springer.

Full Harvard references: blog.bywaleed.com