Chainguard.

All About That Base Image

Vulnerable Base Container Images, Noisy Scanners, and the Case for “Quiet” Images.

John Speed Meyers and Zack Newman

2022

Executive Summary

TL;DR - Using "quiet" base images, minimal images with few or no vulnerabilities and built-in security, can reduce security debt, decrease the developer's workload, and improve development velocity.

Software environments are often referred to as a "stack," layers of software components squashed together. Containers, a modern and popular form of packaging software, take this abstraction literally. Containers are built from "layers," a set of changes to a file system, and are built on top of a so-called base image, sometimes called a parent image, that provides an operating system and useful packages.

The bottom layer of the container stack, the base image, is crucially important. Software teams ought to choose this image wisely because their software application inherits the properties of this base image. Whether these vulnerabilities actually affect the final application or not, these vulnerabilities in a base image amount to security debt, similar to the idea of technical debt, and force developers to either accept the security risks or attempt their own patches and mitigations.

To deal with vulnerabilities found in containers, software teams increasingly use container security scanners, tools that automatically find security vulnerabilities in a container image. But these scanners are noisy, regularly issuing false positives.

To help software developers, security teams, and DevOps professionals better understand the security debt of popular base images, we identified several widely used base images, applied three security scanners to these images, and analyzed the lifetime of these vulnerabilities. Our notable findings include:

- A few base images, such as Alpine, Ubuntu, and Debian, have grown hugely popular.
- Some popular base images can have hundreds of known security vulnerabilities, although the Alpine image, which had zero reported vulnerabilities, suggests that zero known security vulnerabilities for a base image is possible. These findings are not specific to any one scanner.
- Some base images have vulnerabilities that were first reported almost twenty years ago.

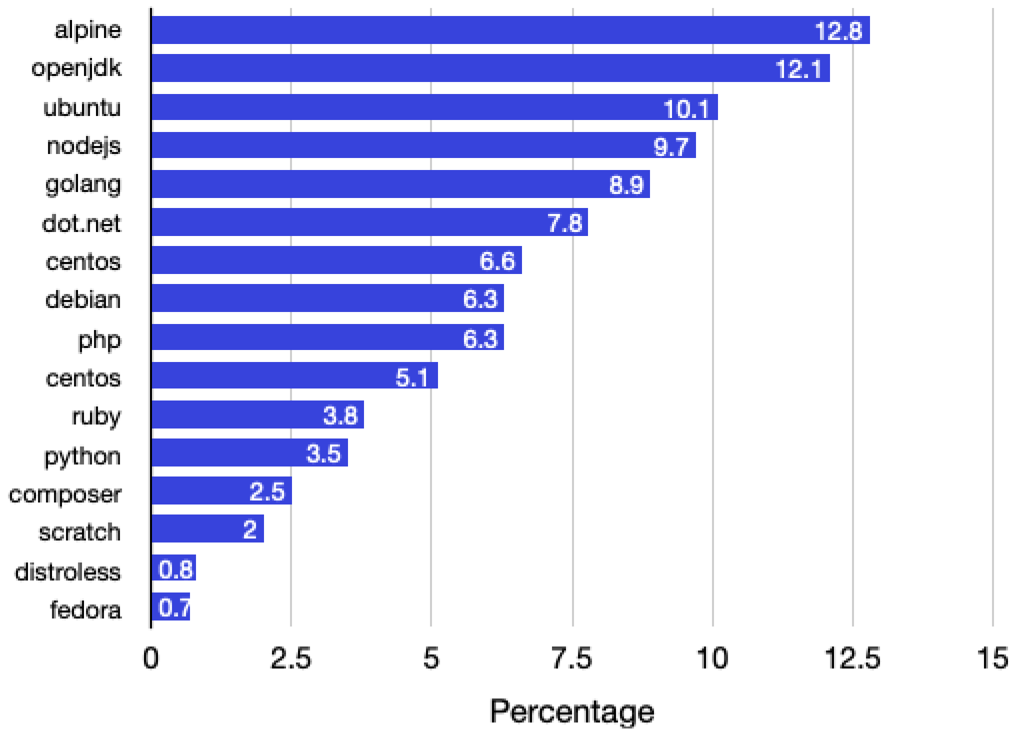
One potential solution to the dual problem of base images with many known vulnerabilities and noisy scanners is to use "quiet" images, minimal images with few or zero known vulnerabilities and additional security features. Fewer reported vulnerabilities in stripped-down base images makes life easier for all parties involved in the software process, improving security and increasing velocity. Adding additional security features such as digital signatures, a software bill of materials, or a service-level agreement on patching timelines offers even more potential. Quiet images, in other words, offer an alternative security experience to developers who are tired of noisy scanners and security-debt-ridden base images.

A Few Base Images Have Grown Immensely Popular

Base images have become the building block of many modern software applications. As a result, many images are now widely downloaded. In February 2022, there were at least nineteen images on Docker Hub that had each been downloaded at least one billion times. Base images are now competing with McDonald's hamburgers (remember "Over 99 Billion Served"?) in the competition for ubiquity.

To understand which base images are most popular, we used GitHub code search to collect data from publicly available "Dockerfile" specifications for container images. Figure one presents the results of a global search across GitHub for all "FROM <x>" commands, the command that imports a base image. This analysis draws on approximately 3.4 million Dockerfiles. To be sure, this data doesn't actually reveal how widely deployed a particular image is, only the number of distinct deployments using each image.

Figure 1. Distribution of All Base Images on GitHub

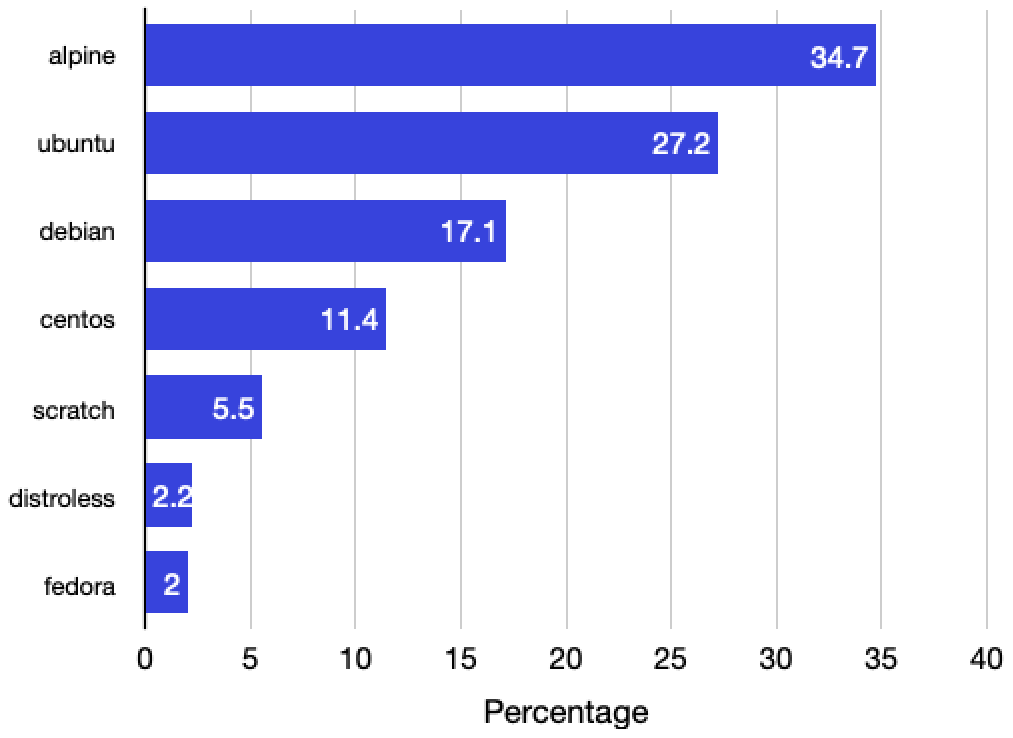


Note: The language-specific base images are sometimes themselves based on other base images. The above analysis does not account for this "base images all the way down" complication. Additionally, this aggregate analysis omits the fact that each image has many different "flavors."

A number of operating system base images such as Ubuntu and Debian are particularly popular, but so are base images that allow the creation of programming language-specific apps like nodejs (for Node/Javascript) or openjdk (for Java).

Filtering on the same data that generated figure one, we also analyzed the distribution of popular base operating system images. See figure two for results.

Figure 2. Distribution of Popular Base Operating System Images on GitHub



Using GitHub code search data, Alpine, Ubuntu and Debian appear to be the most widely used base operating system images.

Some—But Not All—Popular Base Images Have Noteworthy Security Debt

We then picked a mix of base images including several operating system images and the Node image and submitted them to well-known container image static analysis tools. We used three–trivy (0.23.0), grype (0.32.0), and Snyk (0.16.0), which is used by Docker scan–to ensure that this analysis isn't specific to one particular scanner.

We were particularly interested in the number and severity of known security vulnerabilities in a base image. These vulnerabilities can be viewed as security debt given that a software application inherits these vulnerabilities from a base image.

Figure 3 presents a count of total vulnerabilities by using three security scanners for five different base images. These vulnerabilities represent so-called CVEs, or Common Vulnerabilities and Exposures, the term for a widely used database of publicly disclosed cybersecurity vulnerabilities.

Figure 3. Base Image Total Vulnerabilities Count by Image and Scanner

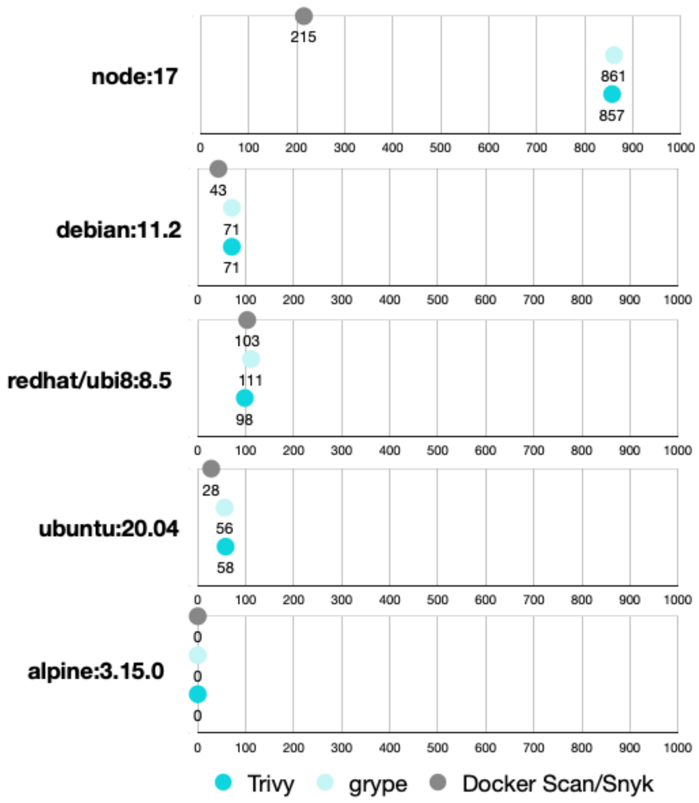


Figure 4 presents a count of vulnerabilities by severity using the same three security scanners for the same five different base images.

Figure 4. Base Image Vulnerabilities Count by Image, Scanner, and Severity

node:17

scanner	# Critical	# High	# Medium	$ Low	# Unknown
trivy	33	122	183	503	16
grype	33	112	163	485	68
Docker Scan/Snyk	13	26	38	138	0

debian:11.2

scanner	# Critical	# High	# Medium	$ Low	# Unknown
trivy	6	4	5	56	0
grype	6	2	2	55	6
Docker Scan/Snyk	3	2	1	37	0

redhat/ubi8:8.5

scanner	# Critical	# High	# Medium	$ Low	# Unknown
trivy	0	0	59	39	0
grype	0	2	65	40	4
Docker Scan/Snyk	0	0	57	46	0

ubuntu:20.04

scanner	# Critical	# High	# Medium	$ Low	# Unknown
trivy	0	0	27	31	0
grype	0	0	25	31	0
Docker Scan/Snyk	0	0	5	23	0

alpine:3.15.0

scanner	# Critical	# High	# Medium	$ Low	# Unknown
trivy	0	0	0	0	0
grype	0	0	0	0	0
Docker Scan/Snyk	0	0	0	0	0

The results suggest that the security-conscious should be wary of the Node (Debian flavor) base image. Depending on the scanner, there are either a couple hundred or closer to a thousand known vulnerabilities in this base image. This is a level of debt that would make even U.S. medical students blush. Notably, scans of the Alpine flavor of node (node:17-alpine) returned between zero and four vulnerabilities, suggesting that the vulnerabilities in the Node base image are associated with the base image itself rather than Node specifically. This finding suggests that the flavor of a base image can have a strong influence on the scan results.

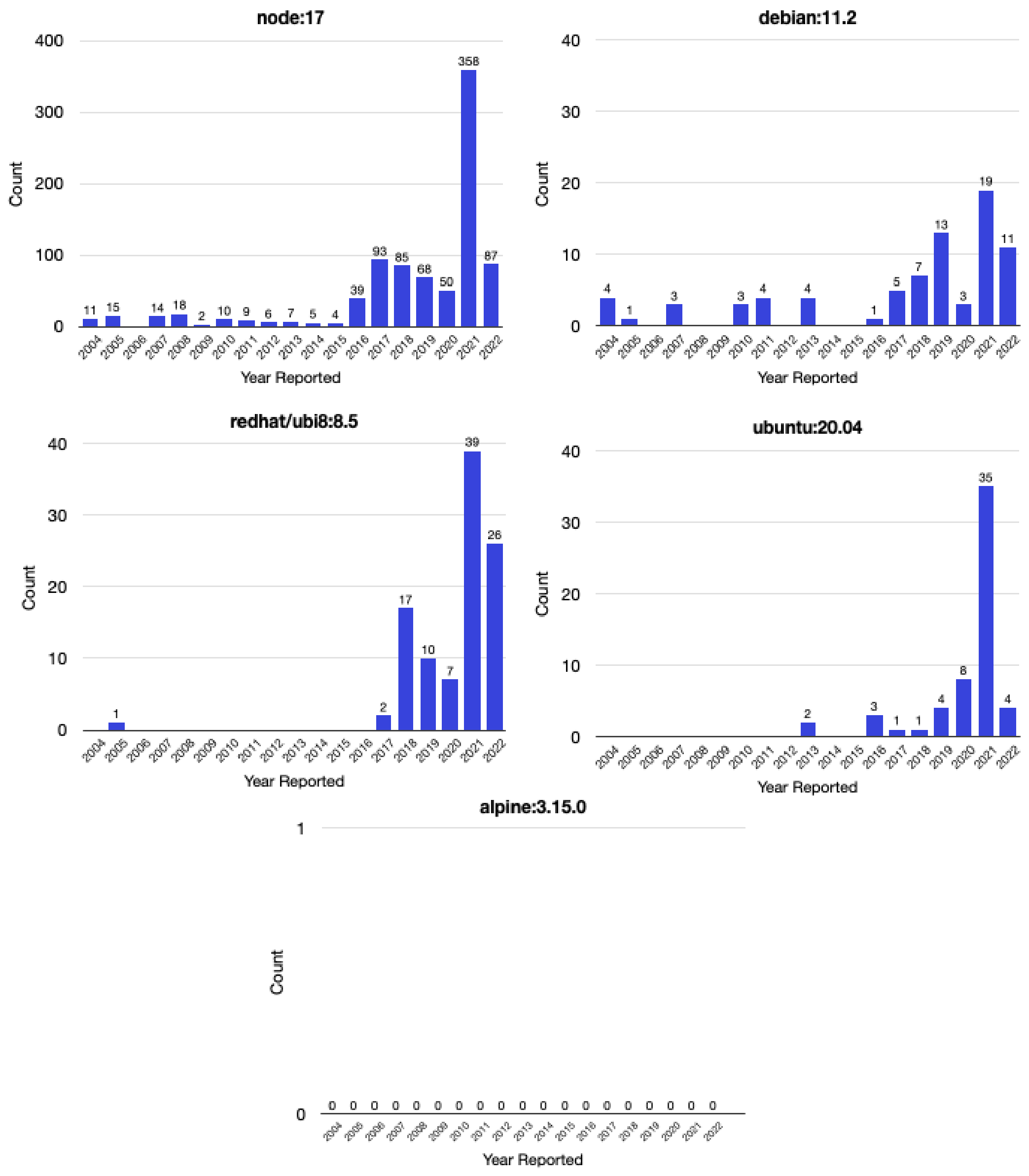
The Debian, Red Hat Universal Base Image and Ubuntu images are roughly similar. There are, depending on the scanner and image, twenty-five to one hundred vulnerabilities and the majority of these vulnerabilities have a medium or low status. This level of security debt is noticeably lower than for Node, though still not zero.

The Alpine base image, a “security-oriented” image that contains less than ten packages, had no known security vulnerabilities in the scanned version. Of course, security vulnerabilities constantly emerge and base images change, so this finding should not be construed as implying that Alpine images are secure forever. It’s also worth noting that the Alpine maintainers have invested time in ensuring the data quality of security fixes and ensuring scanners don’t flag false positives. In sum, zero security debt for a container image is possible.

Paying Off Security Debt is Hard, Or How Old Vulnerabilities Die Hard

To further understand the vulnerabilities associated with these popular base images, we analyzed the year that each vulnerability was first reported for all vulnerabilities within each base image. Figure five presents an analysis for each image using data from the trivy scanner.

Figure 5. Base Image Vulnerabilities by Year Reported for Select Images



Note: Because of the relatively high number of reported vulnerabilities for the Node image, the axis scale for that image was increased by a factor of ten. Additionally, the relatively short bars in 2022 could reflect that these analyses were performed in February 2022.

All base images, except for Alpine, had vulnerabilities stretching back several years. The Node image even had twenty-five vulnerabilities dating back to either 2004 or 2005. Interestingly, the Red Hat and Ubuntu image vulnerabilities were clustered within the past five years while the Debian vulnerabilities were spread across many years. This data suggests that the images do not contain these vulnerabilities due to lack of time. It's more likely that these vulnerabilities remain because of factors such as a conscious decision to not fix a vulnerability and excessive attack surface, the result of extraneous packages, which makes zero-vulnerabilities a herculean task.

In short, security debt can stick with a base image.

“Quiet” Base Images: Avoiding the Din of Noisy Security Scanners

The scanners used for this analysis are likely “noisy,” producing many false positives. Even if these findings are not true positives, however, the many false positives pollute the scan results, forcing software developers and security teams to divert their time from building new software. For instance, high-security organizations such as the U.S. military now require two separate security scans of containers, an onerous requirement especially if there are many reported potential vulnerabilities. Some organizations even devote considerable resources to maintaining long Excel lists of reported vulnerabilities and implemented mitigations.

There are a number of alternative approaches to coping with the dual problem of noisy security scanners and base images with many known vulnerabilities. VEX, the Vulnerability-Exploitability eXchange, for instance, provides a way for software suppliers to use the VEX format to relay the information that a particular vulnerability is not exploitable in the final product. In a similar vein although not focused on containers, OSV.dev, provides an open source software vulnerability scan database and triage infrastructure that provides relatively detailed vulnerability information to ease the vulnerability remediation tasks of open source maintainers and consumers.

Fortunately, there exists another approach, one suggested by the Alpine Linux results. First, base images ideally ought to embrace a “less is more” principle, reducing the image’s attack surface in the same way that tidiness guru Marie Kondo has decluttered our lives. Many vulnerabilities are found in the “clutter” of an image, those extraneous and unnecessary packages along for the ride in a base image. In particular, removing a shell from a base image closes a potentially unnecessary access point for attackers, turning an open door into a brick wall. Second, these minimal base images ought then to have few or no reported vulnerabilities–”quiet” images–which reduces the burden for all involved parties. Developers avoid triaging vulnerabilities. Security teams avoid making long Excel lists of vulnerabilities and remediations. Software users get secure software faster and cheaper.

In short, a quiet base image with security features built-in-by default, like digital signatures, a software bill of materials, and a service-level agreement on patching timelines, offers an alternative approach to the status quo. If you too are interested in peace and quiet when it comes to container base images, please let us know!

References

Ariadne Conill, “The Vulnerability Remediation Lifecycle of Alpine Containers,” Personal Blog, June 8, 2021, available at https://ariadne.space/2021/06/08/the-vulnerability-remediation-lifecycle-of-alpine-containers/.

Drew DeVault, “Developers: Let distros do their job,” Personal Blog, September 27, 2021, available at https://drewdevault.com/2021/09/27/Let-distros-do-their-job.html.

John Speed Meyers is a security data scientist at Chainguard. He contributes to research and data science efforts related to open source software security, advancing Chainguard’s mission of making the world’s software supply chain secure by default.

Zack brings his passions of developer tooling and applied cryptography to Chainguard. After 4 years as a software engineer and tech lead on Google Cloud SDK, he moved to MIT CSAIL to research authenticated data structures and Tor network performance. He is excited to work with the TUF and Sigstore communities to make the open-source world more secure.

John Speed and Zack thank their fellow guardians for the ample constructive critique that improved this research, though all mistakes remain theirs alone.