Over 30% of Official Images in Docker Hub Contain High Priority Security Vulnerabilities

Jayanth Gummaraju, Tarun Desikan, and Yoshio Turner

BanyanOps
{www.banyanops.com, team@banyanops.com}

Docker Hub is a central repository for Docker developers to pull and push container images. We performed a detailed study on Docker Hub images to understand how vulnerable they are to security threats. Surprisingly, we found that more than 30% of official repositories contain images that are highly susceptible to a variety of security attacks (e.g., shellshock, heartbleed, poodle, etc.). For general images (images pushed by docker users, but not explicitly verified by any authority) this number jumps up to ~40% with a sampling error bound of 3%.

In order to perform this study, we pulled images from Docker Hub and inspected packages and their versions installed in them. We then used the information from the Mitre, NVD (National Vulnerability Database) and Linux distro-specific databases to determine which images are vulnerable to attacks. We built an open source tool called Banyan Collector and a service called Banyan Insights to get most of the data needed for this study.

In this paper, we provide a high-level overview of how security vulnerabilities are classified, describe results obtained by analyzing official and general images from Docker Hub for vulnerabilities, and also discuss the implications of our study to operations management.

Security Vulnerability Assignment and Classification

Mitre is a not-for-profit organization that assigns and maintains a list of CVEs (Common Vulnerabilities and Exposures), where each CVE describes vulnerabilities in widely distributed software (e.g., Linux, Windows, etc.). The NVD database maintained by the US government lists the impact for each CVE, including software it affects and the corresponding fix (or fix pending). Each Linux distro (e.g., Debian, Ubuntu, Centos, etc.) also maintains distro-specific impact and the package version that provides a fix for the vulnerability.

Each vulnerability is assigned a score by NVD and Linux Distro. The scores range from 0 to 10, where a score in the range 7.0-10.0 are classified as High vulnerability, those in the range 4.0-6.9 as Medium vulnerability, and 0-3.9 as Low vulnerability. This classification takes several factors into account including the complexity required to exploit a system (lower the complexity, higher the score) and the impact of the vulnerability (higher the impact, higher the score). Some examples are:

- High vulnerabilities: e.g., ShellShock (bash), Heartbleed (OpenSSL), etc.
- Medium vulnerabilities: e.g., Poodle (OpenSSL), etc.
- Low vulnerabilities: e.g., gcc: array memory allocations could cause integer overflow

Note that classification of a vulnerability into High, Medium, and Low buckets is subjective and could be reclassified by companies based on their specific setup. Also, the scores

assigned by NVD could be different from that of each Linux distribution and could change over time. In our study, we used the score assigned by their respective distribution for Ubuntu and Centos; but for Debian we used the scores directly from NVD because we couldn't find a good source for Debian-specific classification. We took a snapshot of Docker Hub and security vulnerability information on 20th May 2015 to perform our analysis. We tried a couple of other days too, and the results were fairly similar.

Evaluation of Official Repositories from Docker Hub

Docker maintains a curated list of official repositories that provides a channel for software vendors or organizations (e.g., Canonical, Debian, Redhat, etc.) to provide up-to-date versions of their container images. They can be identified by their namespace *library* on Docker Hub. Examples of such repositories include *library/ubuntu, library/redis,* etc. Docker Hub contains ~75 official repos (at the time of writing this article), with ~1600 tags referring to ~960 unique images.

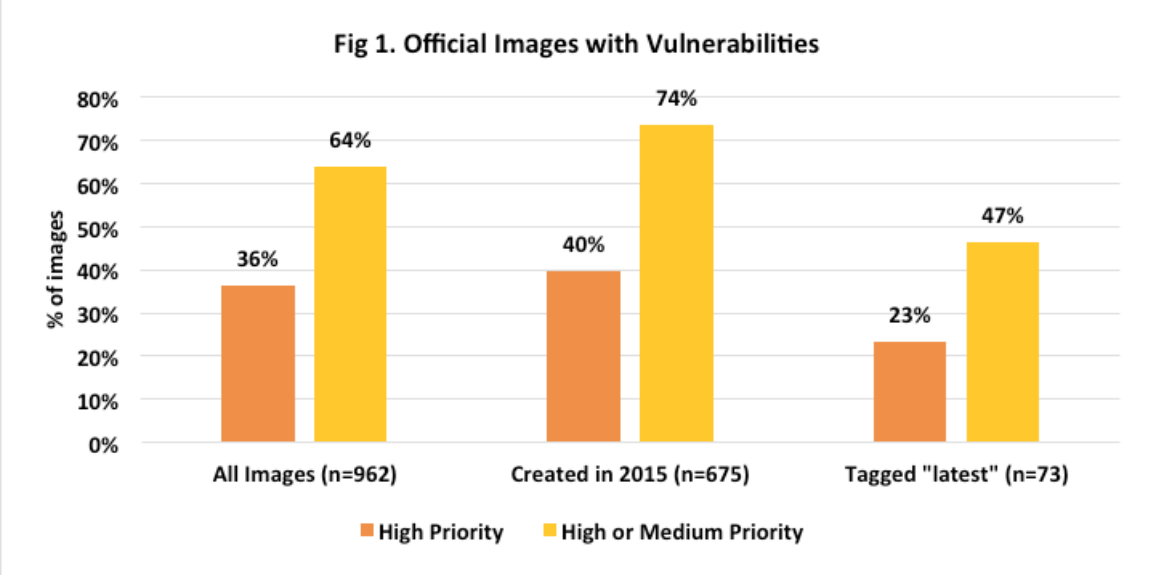


Fig 1. shows the main results obtained by analyzing all official images from Docker Hub. More than a third of all images have high priority vulnerabilities and close to two-thirds have high or medium priority vulnerabilities. These statistics are especially troublesome because these images are also some of the most downloaded images (several of them have hundreds of thousands of downloads).

If we just look at images created this year, the fraction of images with high vulnerability are still over a third, but the high or medium vulnerable images go up close to 75%. In other words, 3 out of every 4 images created this year have vulnerabilities that are relatively easy to exploit with a potentially high impact.

If we limit our scope to images that have the *latest* tag, the fractions go down to 23% and 47% respectively, which are still very significant. The lower numbers for *latest* tag suggest that docker user/maintainers tend to keep their newest images more up-to-date, but the older images get ignored; the sheer volume and velocity of creating containers easily cause older images to be neglected during updates.

To understand the source of these vulnerabilities, especially the high ones, we performed a detailed analysis of packages affecting most of the Docker Hub images.

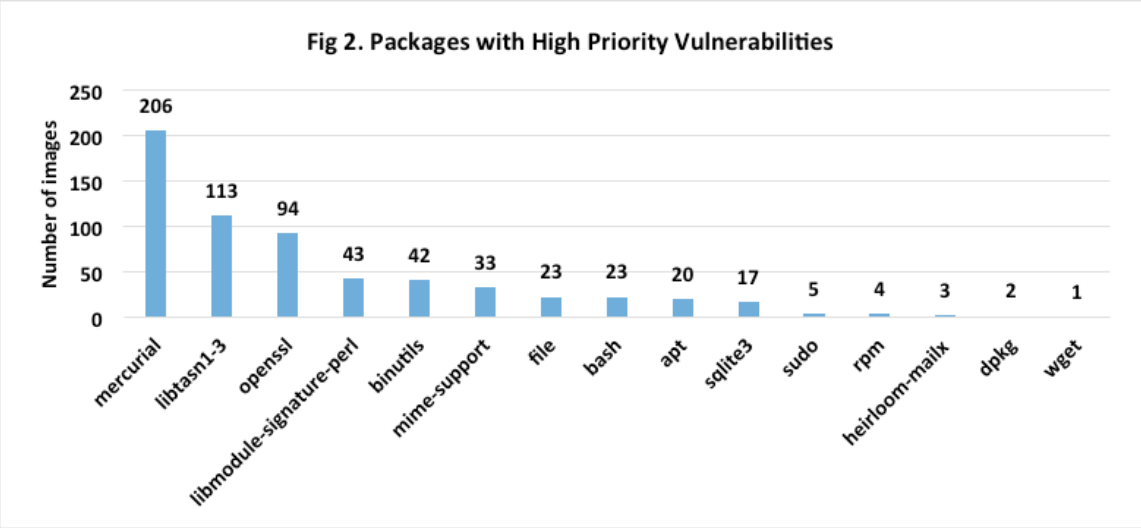


Fig 2. showcases the main results of this analysis, and Table 1. lists the main CVEs associated with these packages. The recently released vulnerability in mercurial is present in a large fraction of images (~20%). High-profile OpenSSL vulnerabilities such as Heartbleed and Poodle are present in close to 10% of official Docker Hub images! Some of the images also contain bash ShellShock (e.g., Centos 5.11), which was discovered more than 7 months back. Even if organizations don't use some of these packages, not explicitly removing them from containers makes them highly vulnerable to malicious attacks.

Package	High Priority CVEs
bash	CVE-2014-6271,CVE-2014-7169,CVE-2014-7186,CVE-2014-7187
apt	CVE-2012-0954,CVE-2012-3587,CVE-2013-1051,CVE-2014-0487,CVE-2014-0488,CVE-2014-0489,CVE-2014-0490
openssl	CVE-2014-0160,CVE-2014-3512,CVE-2014-3513,CVE-2014-3567,CVE-2015-0292
libtasn1-3	CVE-2015-2806
file	CVE-2014-9653
mime-support	CVE-2014-7209
binutils	CVE-2014-8485,CVE-2014-8501,CVE-2014-8502,CVE-2014-8503,CVE-2014-8504
mercurial	CVE-2014-9462
rpm	CVE-2013-6435,CVE-2014-8118
heirloom-mailx	CVE-2004-2771
sqlite3	CVE-2015-3414,CVE-2015-3415,CVE-2015-3416
ntp	CVE-2014-9293,CVE-2014-9294,CVE-2014-9295
sudo	CVE-2013-1775
libmodule-signature-perl	CVE-2015-3408,CVE-2015-3409
httpd	CVE-2012-5575,CVE-2015-0226
wget	CVE-2014-4877

Evaluation of General Repositories from Docker Hub

In addition to a handful of official repositories, Docker Hub contains a large number of general repositories (~95,000 at the time of writing this article) and hundreds of thousands of unique images. For our experiments, we selected ~1,700 images at random and analyzed their contents (error bound ~3%).

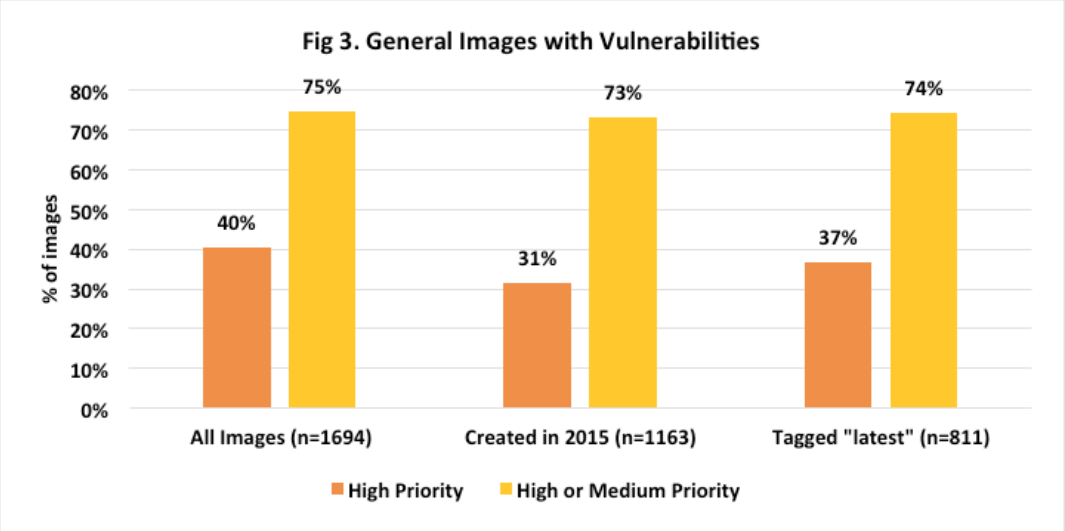
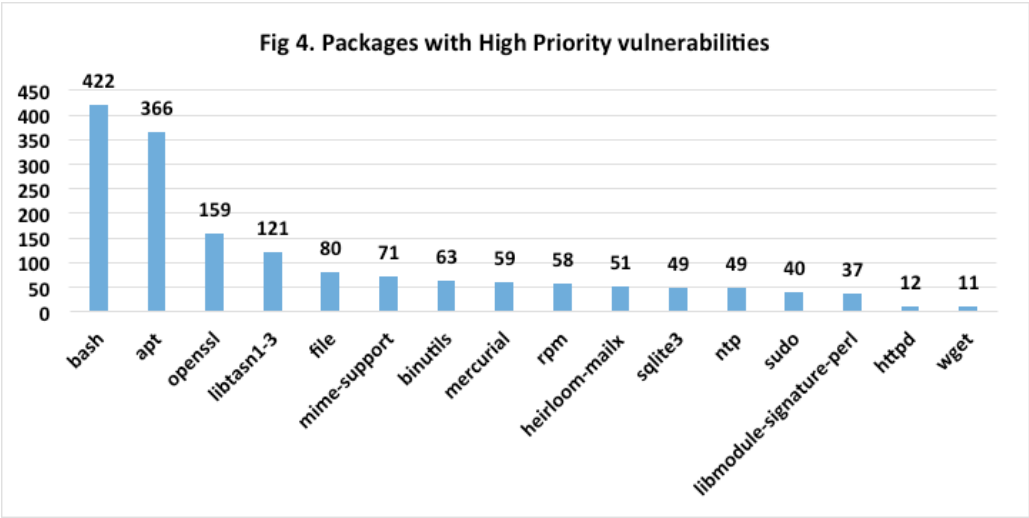


Figure 3. shows the main results upon analyzing general images. Overall, the percentage of vulnerabilities is significantly higher than that of Official images. This result is expected because there is no process in place to sanitize general images before pushing them to Docker Hub.

Almost 40% of general images have high priority vulnerabilities. Even if we just look at images created this year or ones with the *latest* tag, the percentage of vulnerable images hovers between 30-40%. If we include images that have medium vulnerability, this number jumps up to more than 70% for all the time frames. Although one might argue that these images have far fewer number of downloads compared to Official images, the sheer number of them (several hundred thousand images) indicate that they are prevalent almost as much as Official images.

We again looked at the breakdown of packages that affect most general images only considering High priority vulnerabilities. Figure 4. summarizes the key results.



Interestingly, unlike Official images where mercurial was the most prolific package affecting images, in these images openssl, bash, and apt are the most common packages. We suspect that the primary difference in the official and general numbers stem from the difference in Linux distributions that constitute the bulk of these images. Official images are typically built on Debian and a large fraction of them have the mercurial package. General images, on the other hand, seem to be more commonly built on Ubuntu and these have bash, apt, and/or openssl related vulnerabilities.

Implications

Containers have revolutionized software development by providing a very efficient path to take software written by developers and run it in production in a matter of minutes or hours, rather than days or months using traditional approaches. But our data shows that these benefits come with a caveat -- without sound operations and security management practices, we are potentially making our software ecosystem more susceptible to attacks.

Containers provide a layer of isolation between applications in separate containers, thereby increasing security. Containers still need to communicate with other containers and systems, however, and thus they remain susceptible to remote exploits of security vulnerabilities baked into the container images, such as those uncovered by our analysis. Moreover, the great ease and speed of portably launching large numbers of containers in diverse environments, e.g., public cloud, private cloud, laptops, make it challenging to track down and secure containers that have vulnerabilities. The high velocity of container deployment greatly boosts the churn and diversity of deployed software, accelerating the introduction of new vulnerabilities into environments.

Another fundamental aspect of using containers is that package management has moved inside containers from the traditional approach where packages were managed in only a base OS installed on Virtual (or Physical) Machines. This shift mainly stems from the difference in abstractions provided by Virtual Machines (VMs) vs. Containers. VMs provide a machine-centric abstraction, which is typically long-running, contains packages to support multiple types of applications, and is kept secure by constantly patching the packages.

In contrast, containers provide a more process-centric abstraction of an ephemeral, portable, and immutable entity where only the packages needed for running a single application are baked into a container. Any new updates require rebuilding the container images to maintain immutability. The packages also get replicated in multiple containers, causing any vulnerabilities to also get replicated.

Furthermore, the shift to a new Devops model where developers are responsible for packages needed by their applications implies that the developers now take on the additional onus of maintaining packages. In addition to operating system packages, developers can include application-level modules in container images using package managers like pip (python), npm (node), and maven (java), which were outside the scope of our study, but can also introduce vulnerabilities. Since developers are more focused on getting new features out as fast as possible, it's fairly easy to neglect keeping older images up-to-date, as our results have also demonstrated (e.g. the official Centos 5.11 image built in April 2015 still has the shellshock vulnerability reported over 8 months prior, in Sep 2014!).

A great way to avoid some of these issues is to rebuild images with the latest updates on a regular basis. The rebuild process must use the latest base image provided by a distributor, and cannot use any cached image layers (e.g., using *--no-cache* in addition to *apt-get upgrade*). But rebuilding from scratch and redeploying all the containers any time a vulnerability is found has huge overhead and is just not practical -- vulnerabilities surface very frequently (multiple times a day) and it's hard to estimate the potential impact of each one. Moreover, updating the containers' packages could easily induce side-effects and instability in developer applications that may go unnoticed even by using sophisticated tests, making it even less desirable to rebuild often.

Conclusion

Our findings advocate a rigorous operations management process where images are analyzed in real-time to provide full visibility into their contents. The images should be scanned for security vulnerabilities, and selectively marked for rebuild depending on the relevance and severity of the vulnerabilities. Any major vulnerability should be identified instantly and there should be an option to trigger an immediate quarantine of susceptible images. The images not only need to be scanned for OS-level package vulnerabilities, but also application-level package vulnerabilities. These processes need to be efficiently integrated into a continuous deployment framework to realize the full benefits of containers while simultaneously maintaining good security practices.

