aqua

Whitepaper:

Blueprint for Vulnerability Management in The Container Pipeline

Last Updated: Dec. 2016 | v1.0

The Challenge: Managing Vulnerabilities in Container Images

One of the most salient security challenges in using Docker containers is the assessment and management of vulnerabilities in container images, during their development and deployment lifecycle.

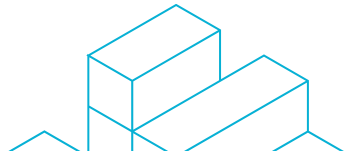
While there are plenty of tools to manage container development pipelines, and several tools that can scan Docker images for vulnerabilities, managing this process in a smart, efficient and effective way presents an elusive challenge due to several factors that make it different from traditional vulnerability management.

Ultimately, the blueprint for a workable, optimized solution must include several elements that work at all stages of the pipeline, including the ability to curb base image use, reporting only applicable CVEs, providing actionable insight to developers, and integrating with development tools such as CI/CD to automate the process.

How Are Containers Different?

To understand why containers require a somewhat updated approach to vulnerability management, we must understand what differences they introduce when compared with traditional development environments, even agile ones:

- **Reliance on Base Images:** When developers build a Docker image, the don't usually start from a clean slate. They start from a base image, often for an OS (such as Ubuntu or Alpine) or a common application (such as Nginx or JBoss). Sometimes the base image used will in itself be based on other base images. This means that vulnerabilities detected in the image may actually originate in a base image, or even in that base image's base image. Remediation should optimally occur at the point of origin.
- **Different Approach to Patching:** Whereas traditional applications are often patched in their runtime environments, containerized applications require patching to be performed in the image. The containers themselves, which are immutable, are then simply replaced with new ones instantiated from the fixed image.
- **Rapid Cycles, Continuous Integration:** In CI environments using containers, the most convenient way to commit new code is to trigger a Docker build and create a new image. This means that images are created daily by each developer, and in large teams this may translate to thousands of new images being built every week or every day.
- **Code + Components + Binaries:** Container images may contain code, packaged components, and compiled binaries - all in a single image. This smorgasbord of

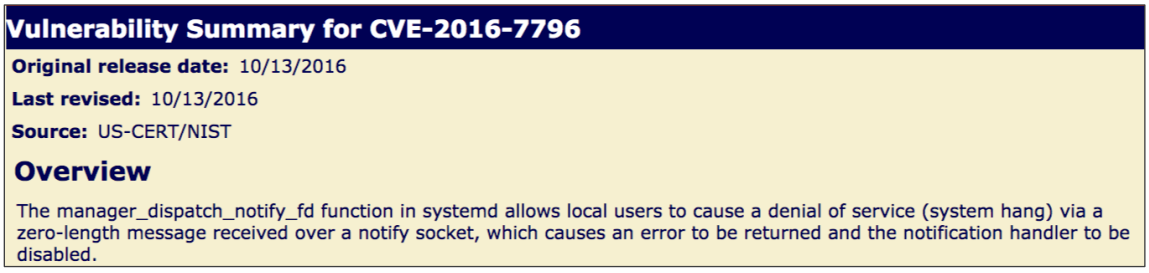


different elements makes it difficult for scanners that rely on specific technologies to provide a complete view of the image contents.

What Do Developers Need?

In container environments, more so than with traditional (slower) development environments, developers hold the key to managing vulnerabilities. This is due to both the rapid pace at which container images are updated and the short release cycles (sometimes daily or even several times a day), as well as the reliance of developers on open-source components in addition to their own written code. Therefore, a vulnerability management solution for containers must be one that developers find useful and easy to work with. But what do developers need?

- **Actionable Insights:** Developers are not security researchers. They cannot be expected to figure out the relevance, applicability and location of a vulnerability within the code, nor can they be expected to understand how to remediate the vulnerability without guidance. What they need is actionable insights that explain what should be updated, patched or fixed.



Vulnerability Summary for CVE-2016-7796

Original release date: 10/13/2016

Last revised: 10/13/2016

Source: US-CERT/NIST

Overview

The manager_dispatch_notify_fd function in systemd allows local users to cause a denial of service (system hang) via a zero-length message received over a notify socket, which causes an error to be returned and the notification handler to be disabled.

How can a developer act on the above CVE description from NVD?

- **To Not Be Overwhelmed:** Many CVE scanners can find a multitude of vulnerabilities, but only some of those would require remediation within a given context. Other CVEs might be risky yet but represent very low risk, while others yet might be completely negligible. When a developer is given a report with 50 CVEs, with only 3 CVEs that really matter, he would feel justifiably overwhelmed and will probably not get to fix those 3 CVEs. A more pragmatic approach is needed here that takes the 80/20 rule and puts it into practice.

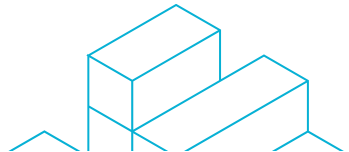
In the above example a scanner found a total of 344 vulnerabilities, 83 of them rated as high severity – which would be off-putting to any developer, and cause them to waste time on fixing vulnerabilities that may be negligible in their environment.

- **"My Code" vs. Pre-Packaged Code:** When creating Docker images, developers often start from a base image, which may already have some CVEs to begin with. When looking at CVEs, developers must be able to tell which CVEs were added in the code that they added to the image, as opposed to CVEs originating in the base images.
- **Scanning Speed:** Developers of containers work in CI environments, and speed is crucial. If vulnerability scanning unreasonably slows down the process, or requires manual initiation, it will not be used for long. The entire vulnerability management process must be tightly integrated into the CI environment, allowing developers to use their familiar tools, and not adding significant time to the cycle.
- **Scalability:** While a single developer may not care about this, R&D organizations would. If your team includes hundreds or thousands of developers, and they all commit code daily, the resulting container image pipeline is quite massive, and requires a highly scalable solution for vulnerability management. Additionally, there may be spikes that require thousands of images to be scanned in minutes - and this is already happening in some customer environments.

Where Do Generic Security Solutions Fall Short?

There are many solutions that offer vulnerability data and the ability to detect and remediate vulnerabilities in software - code scanners, vulnerability assessment tools, etc. However, they may have shortcomings that make them ineffective for container deployments:

- **Separate solutions for static code scanning and runtime scanning:** Since container images may contain both procedural language code, as well as components and binaries, the solution must be able to scan all of them and provide a coherent report. Most traditional tools provide either static or dynamic scanning, not both.
- **Understanding Image Hierarchy:** Images may have nested layers of base images, that impart their vulnerabilities on the final image. If the hierarchy is not understood, it would be difficult if not impossible to address the vulnerabilities originating in base images.

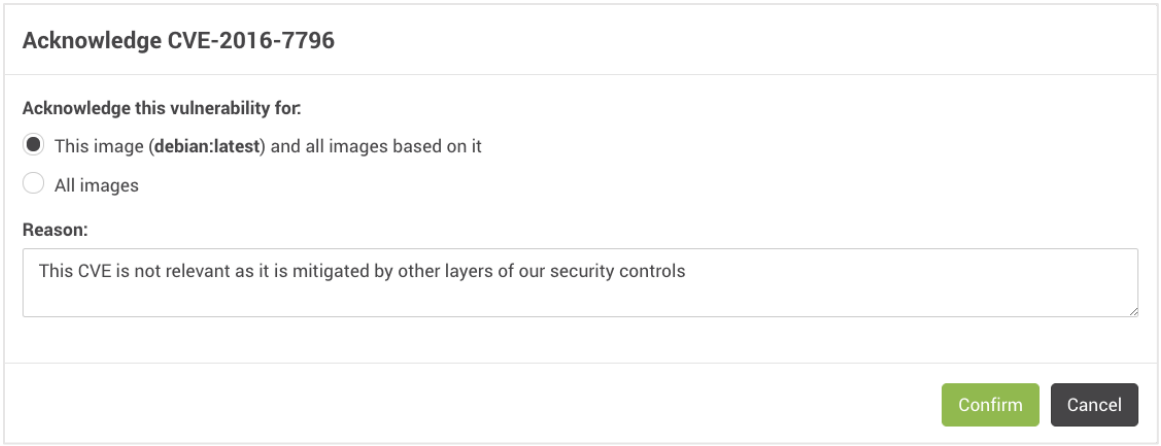


- **Handling Image Pipeline Speed and Scale:** Traditional tools were created for traditional software development speeds, and are usually only used before version releases, i.e. several times a year. They are not built for environments that require scanning several times a day.

The Smart Approach: Optimize Across Lifecycle Stages

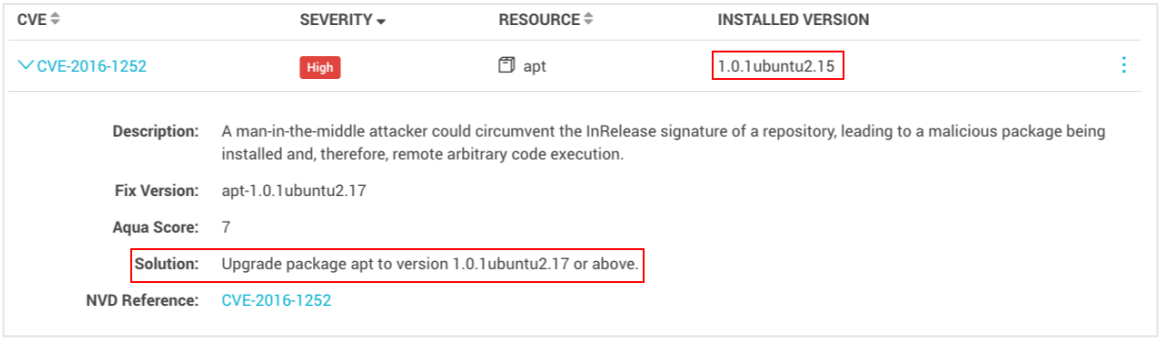
Due to the massive scale and very rapid speed in which container images are developed, updated, and deployed, the right approach to managing vulnerabilities in the pipeline must be pragmatic. It must introduce at every stage of the process the right controls that minimize vulnerabilities in the pipeline, and automate this process as follows:

- **Control the Base Image Inflow:** As noted above, base images can be the source of many vulnerabilities (and much frustration…), and this issue can become exponentially more complex if the choice of base images is unlimited and unchecked. The solution should start by controlling this inflow, whitelisting the base images that can be used, and restricting the sources from which they can be obtained. This must also be enforced in a way that does not allow developers or admins to bypass the process by running containers from images directly on hosts.
- **Report Only *Applicable* CVEs**: Organizations should be allowed to set thresholds for negligible CVEs that will not be sent back to developers. This mechanism can be based on the CVE score, its severity level, and whether it appears in the relevant context. The should also be a CVE whitelist, to ensure that the same negligible CVE is not flagged again and again.



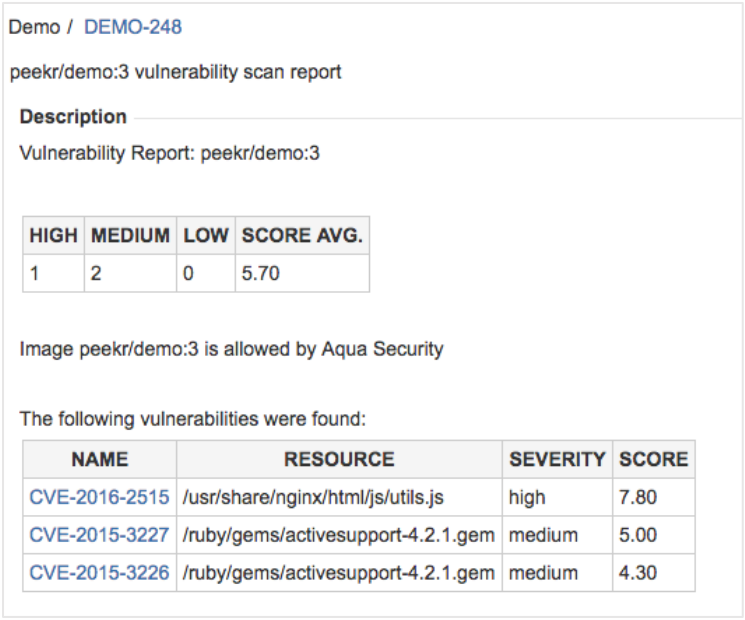
Acknowledging a CVE will prevent it from creating unnecessary noise in future scans

- **Produce Actionable Results:** Since, ultimately, it is up to developers to remediate vulnerabilities that were found in the code they supplied, they must be given reports that they can act upon. Dry CVE data is usually not sufficient, and would require security staff and developers to put their heads together to understand how to act on it, something that is both time consuming and a source of unnecessary friction. Instead, it's better where possible to translate the CVE data into actionable insights that developers can act upon.



Offering clear guidance for vulnerability remediation

- **Close the Loop with Dev Tools:** Perhaps the cherry on top of a developer-friendly solution is the ability to allow developers to implement vulnerability management in the environments that they use for their work anyway. For example, open a ticket in Jira to fix a vulnerable image, or send a message on a Slack channel for the relevant team/project.



Jira screen showing scan results being fed back to developers

* * *

Aqua enables enterprises to secure their virtual container environments from development to production, accelerating container adoption and bridging the gap between DevOps and IT security.

The Aqua Container Security Platform provides full visibility into container activity, allowing organizations to detect and prevent suspicious activity and attacks, providing transparent, automated security while helping to enforce policy and simplify regulatory compliance.

For more information, visit www.aquasec.com

aqua