7 Tips for prioritizing application and container vulnerabilities

Fixing each and every vulnerability is impossible, and most likely - not necessary. You have to prioritize. But where do you start?

1. CWE

Common Weakness Enumeration (CWE) is an online glossary categorizing weaknesses into different types. It can used for prioritization in a number of ways:

- Short on time? Use the **CWE Top 25** list as a starting point
- **Prioritize by impact** and focus on the CWEs most dangerous for your organization
- Use CWSS and CWRAF - two scoring systems for CWEs that allow you to **prioritize based on relevancy to your business**

2. CVSS/Severity

The Common Vulnerability Scoring System (CVSS) is an industry-standard for assessing the severity of vulnerabilities. Based on how easily a vulnerability can be exploited and the level of impact if an exploit were to occur, CVSS assigns a score between 0 and 10, 10 being the most severe score that can be assigned.

According to the NVD, a CVSS (v3) base score of 0.0-3.9 is considered "Low" severity; a base CVSS score of 4.0-6.9 is "Medium" severity, and a base score of 7.0-8.9 is "High" severity, and a base score of 9.0-10.0 is “Critical” severity.

While all vulnerabilities are important, start with those assigned a high and critical severity level.

3. Exploitability

The availability of exploit code for a vulnerability makes it easier for hackers to attack. Once such code is published, this is known as an “exploit in the wild”. **Prioritize those vulnerabilities you know have exploits.**

Distinguishing between the different types of published exploits will help you narrow your focus even further - there is a difference between the risk posed by an exploit that is mature, published and practical and an exploit that is academic and theoretical.

4. Reachability

Some vulnerabilities might not be called as part of the application’s execution path and pose less of a risk. **Rather than prioritizing a high-severity vulnerability in a function not called by your application, it is much more effective to prioritize a medium-severity vulnerability that lies directly in the execution path.**

You can prioritize with either, or both, of the following complementary approaches:

- Determining reachability with static analysis of your source code
- Determining whether vulnerabilities are actually being called in runtime

Important! This does not mean vulnerabilities not being reached are not important. They also need to be triaged and fixed, especially the high-severity ones.

5. Age

Try and factor in the age of a vulnerability when making your prioritization decision. The more recent a vulnerability, the less of a chance of an available fix. This fact also attracts hackers and therefore makes newer vulnerabilities more risky

6. Fixability

If speed is of the essence and you are trying to fix as many issues as possible within a short timeframe, consider prioritizing the issues that are the easiest, quickest, and cheapest to fix.

To do this, try and prioritize by answering these questions:

- Is a fix available? If so, what does it entail?
- Will the fix introduce new issues or break your code?
- Is there a patch available for the vulnerability?

7. Automation

The only way to prioritize and de-prioritize hundreds, if not thousands, of issues across the different projects in your organization is with the help of automation. **Use policies,to automatically set the parameters guiding prioritization decisions and your fix efforts, at scale.**