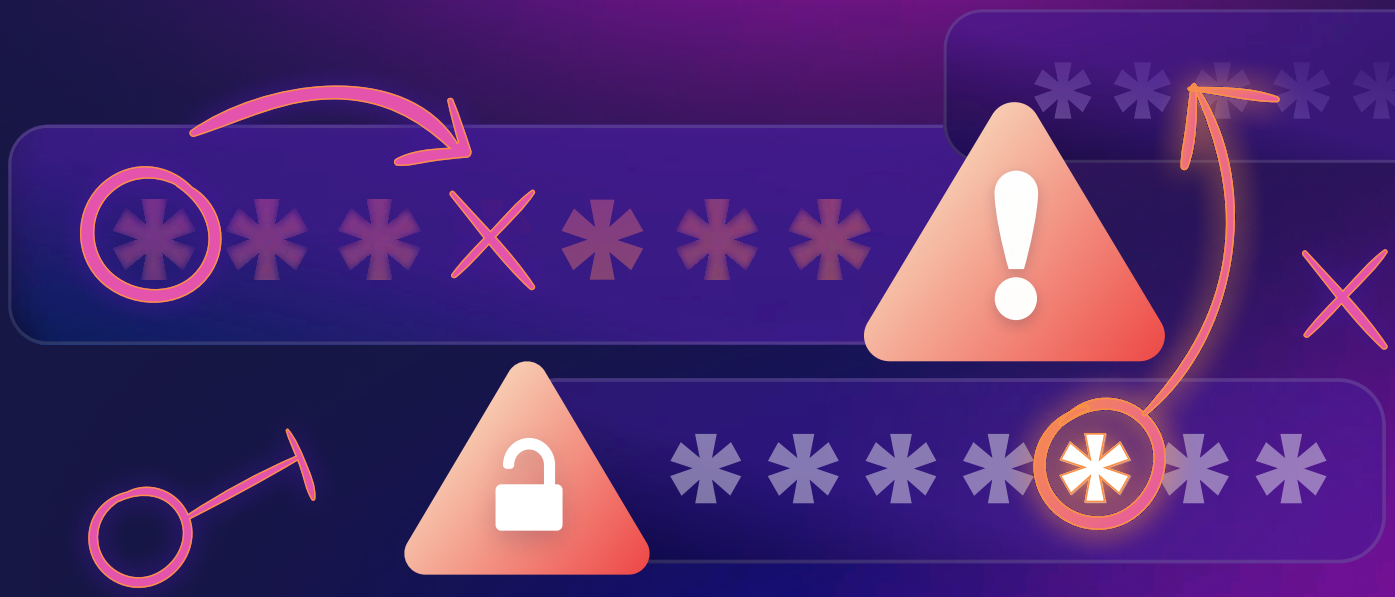
snyk

ZERO-DAY VULNERABILITY PLAYBOOK



How to prepare for tomorrow's zero-day vulnerabilities today

Your playbook for responding quickly and effectively to new zero-day vulns

Conditions can — and often do — change overnight in the world of software security. One day, your applications and systems will be running safely in the cloud. The next, a new exploitable vulnerability hits the headlines and it becomes a race against time to find and fix this vulnerable component across your environment.

Zero-day vulnerabilities — newly found security issues, often without remediation solutions other than "stop using it for now" — throw security teams into a frenzy like nothing else. Fortunately, organizations can take proactive steps to prepare for zero-day vulnerabilities before they even happen (or are discovered).

In this guide to zero-day vulnerabilities, we cover the basics of zero-days and then provide a playbook that your team can use to prepare for any zero-days on the horizon.

What are the most recent zero-day attacks?

When we think about recent zero-day exploits, the first significant example that comes to mind is Log4Shell: a remote code execution vulnerability in Log4j impacting Java applications identified with a CVSS rating of 10 (the highest possible score). This vulnerability affected most major enterprise businesses, as many used Log4j across their environments, often via indirect dependencies.

Most recently, the Shai-Hulud NPM supply chain incident served as another stark reminder of both the urgency to address zero-day exploits as proactively as possible, as well as the overal risk of increasingly complex software supply chains.

A few other examples of impactful zero-day incidents include:

- CVE-2024-47176: a zero-day remote code execution vulnerability found in the CUPS printing system
- CVE-2024-21626: A bug discovered by Snyk in runc that allows a container to escape by manipulating the working directory during a docker run or docker build command
- CVE-2024-23651: A race condition in BuildKit's mount cache handling that could lead to a container escape
- CVE-2024-23652: A flaw that could allow an attacker to delete arbitrary files on the host system during the container teardown process
- CVE-2023-44487: A high-severity HTTP/2 rapid reset zero-day vulnerability.
- CVE-2023-38545: A high-severity, heap-based buffer overflow in cURL
- CVE-2023-4863: A critical vulnerability that allows bad actors to exploit Google Chrome with maliciously-formed WebP images

All of the high/critical severity vulnerabilities in the above list, among others, emerged within a 24-month period, highlighting how often major flaws come to light. In other words, this isn't an issue that security teams can ignore.

Why do zero-day vulnerabilities pose a risk?

Most of today's applications contain a massive network of dependencies and external resources. A single application can include dozens of open-source components and pieces of third-party software. When the general public discovers a zero-day vulnerability, the organizations using the vulnerable component must locate and fix it as soon as possible. Organizations that don't beat the clock and fix these vulnerabilities can fall victim to a zero-day attack.

Lifecycle of a zero-day

Zero-day vulnerabilities are especially risky because they happen so quickly, and their lifecycle can vary considerably, depending on who discovers them. They can be unknown when you go to bed, then all over news headlines and forums when you wake up the following day. Here's a possible timeline of a typical zero-day vulnerability in an open source library (but a similar flow can occur with first-party applications):

1. **A developer selects an open source library.** The updated code and new library get released to production
2. **Vulnerability discovery.** Security researchers (best case) or bad actors (worst case) find the vulnerability over the following days, months, or years. Bad actors start to exploit the vulnerability while security researchers begin digging into how the vulnerability could get exploited.
3. **Vulnerability awareness**. The maintainers learn about the vulnerability but don't have a fix for it yet.
4. **Vulnerability disclosure.** Either the maintainer of the vulnerable library or the security researchers that identified it release information on the vulnerability to affected customers or the general public.
5. **Zero-day patching.** The vendor releases remediation solutions, often as point releases (possibly more than one, depending on how many releases the vulnerability has been around through). The time to release a complete fix varies from hours to a few months, depending on the complexity.
6. **Customer deployment of fixes.** Lastly, it's up to the affected customers to deploy the patches. This process can take months or even years, depending on how long it takes the users to discover every instance of the vulnerable component and deploy the fix.
7. **Continued attack attempts.** Well after a patch is released, bad actors will continue to attempt to exploit the vulnerability, hoping to find an unpatched system in the wild. This is why organizations should always make regular security scanning a part of the build and deploy process.

There can be many steps in between, and, depending on whether or not the teams involved follow best-practice security disclosure policies, can make a big difference on the impact of a given vulnerability.

Playbook: 6 Tips to be proactive planning for zero-day vulnerabilities

Zero-day vulnerabilities happen frequently, leaving thousands of organizations in the lurch every year. In 2022, Google recorded 41 zero-days in the wild. So, it's not a matter of if but when your organization will face a zero-day vulnerability.

Because these vulnerabilities happen so frequently, organizations must proactively prepare for them. Enterprises can take a few steps in a time of non-emergency so they are ready to jump into action and resolve zero-day threats in an emergency. These steps include:

1. Follow a security-first mindset during development.

A security-first mindset means developers are fixing vulnerabilities as they arise throughout the SDLC. Remediating security issues as they happen is far more efficient than waiting days or weeks to correct a lengthy list of issues that have traveled downstream.

A security-first mindset also puts the security team in a better position to collaborate with, rather than frustrate, the developers. This level of collaboration leads to a more robust DevSecOps culture. When the development and security teams are already used to working together, they can solve problems more effectively when a zero-day vulnerability happens.

2. Set up a robust security process.

Robust, shift-left application security processes focus on finding and fixing vulnerabilities throughout the SDLC rather than waiting until the end of each big milestone to test. For example, some organizations conduct static application security testing (SAST) on first-party code whenever a developer creates a pull request.

These processes also include locating all third-party dependencies with software composition analysis (SCA) and then keeping a detailed record of them with a software bill of materials (SBOM). Because today's development processes are so fast-paced, many organizations leverage automation tools to continually track their dependencies in real-time.

When an organization already leverages robust security processes daily, it's far easier to locate projects leveraging third-party resources compromised by zero-day flaws, allowing you to fix them quickly.

3. Perform regular security assessments and audits.

It's also essential for organizations to perform security assessments and audits regularly. While the specifics of these assessments vary by industry and other business needs, they often use some form of these five stages:

- Determining potential threats
- Identifying sensitive data in the environment
- Mapping the attack surface
- Evaluating any process pain points
- Building a roadmap for refining the process

By regularly performing these steps, an organization can ensure that its security program is in tip-top shape. Assessments also help the organization maintain a deep understanding of its existing security posture. This knowledge and preparation makes preventing a zero-day exploit much easier.

4. Facilitate employee awareness.

Employee awareness plays a role in preventing zero-day vulnerabilities as well. An effective training program educates developers on secure code practices in an interactive, engaging way. One of the best ways to educate development teams is by including remediation guidance alongside vulnerability scanning. For instance, some scanners explain why each vulnerability is a risk and how to remediate it.

Hands-on security education pushes developers to understand their own code and its potential security risks better, making them more efficient at resolving zero-day vulnerabilities when they arise.

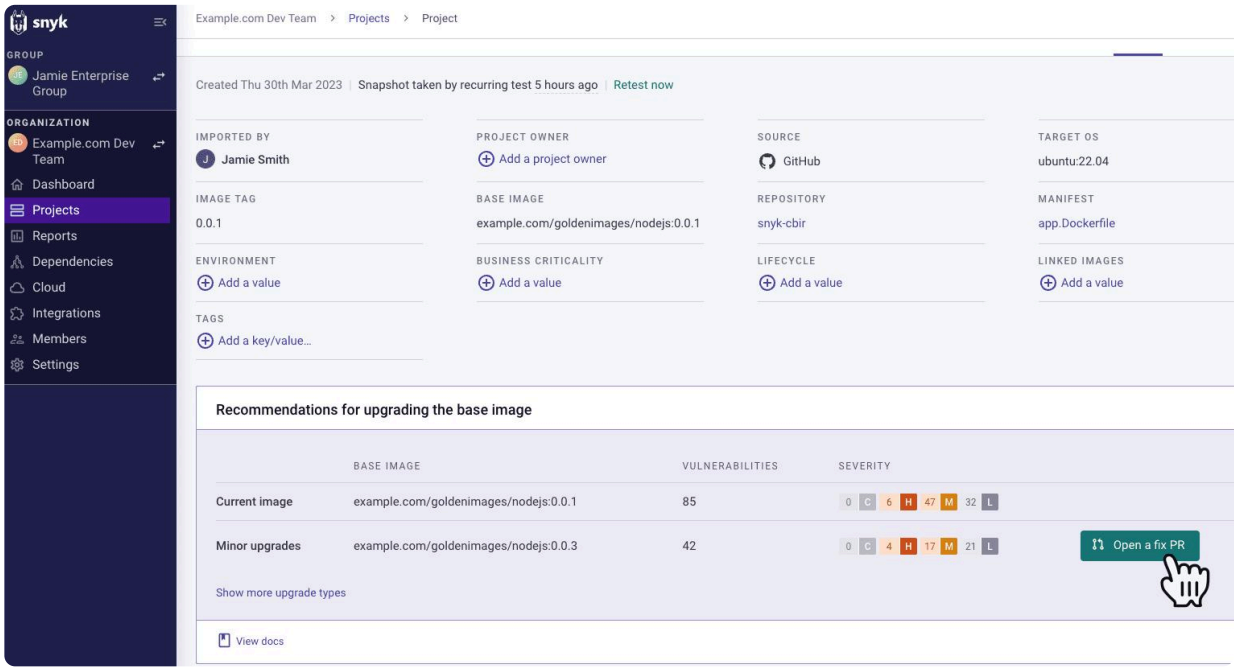
5. Create your own playbook.

Don't wait for the next zero-day vulnerability to document your own processes. Create playbooks and checklists for how to handle the next vulnerability, including which people to involve, how to assess impact, the inventory that needs to be evaluated, and how you'll communicate findings and impact internally (and externally if applicable). Perhaps most importantly, make sure that people know where to find the playbooks. Before the next critical vulnerability, start to incorporate tools to track your source code and container inventory so you can be more prepared for the next vuln.

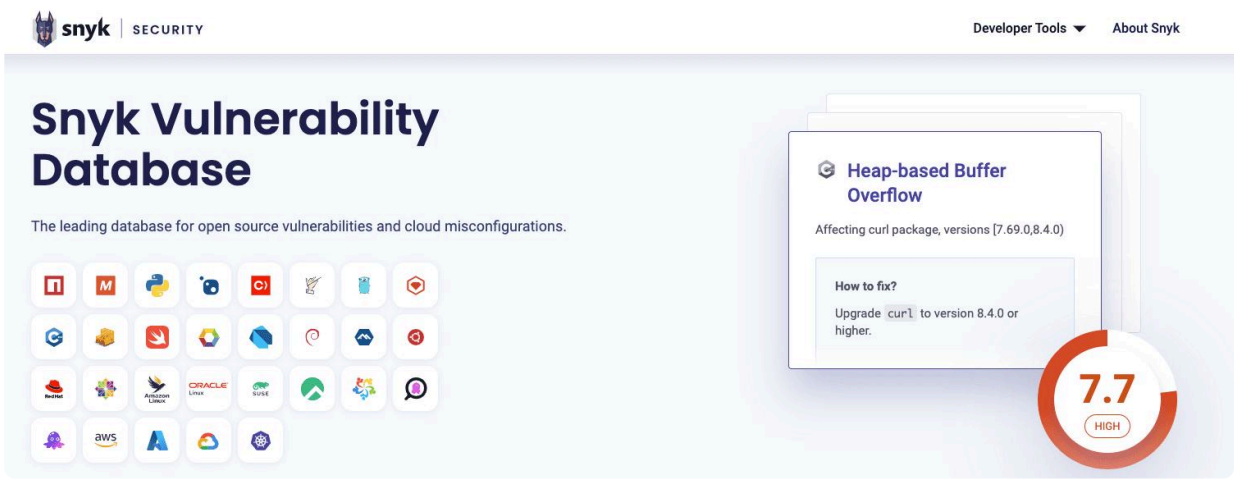
6. Stay up-to-date with the latest security intelligence.

Your organization must also decide on a resource for staying up-to-date on new zero-day vulnerabilities. Time is precious when a zero-day hits the news, so your teams should prepare, so you can quickly understand where to focus fix efforts ahead of time.

Snyk, backed by the Snyk Vulnerability Database (VulnDB), provides detailed information on all ongoing vulnerabilities and explains how to resolve them — often with a click. The sooner your team has the proper information on fixing a vulnerability, the faster you can mitigate or patch it.

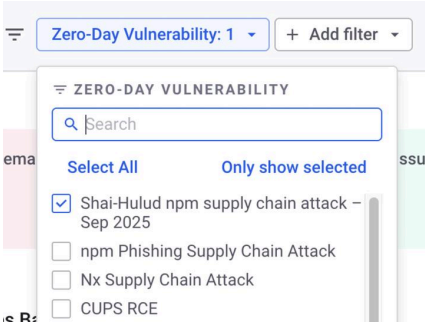


Additionally, you can just go straight to VulnDB to see the latest trending vulnerabilities:



How developer-first AppSec tools can prepare you for the next zero-day

Having the right tools enables teams to execute collaborative, efficient responses to zero-day vulnerabilities. Application security solutions such as Snyk Code, Snyk Open Source, and Snyk Container empower development and security teams to find and fix zero-day vulnerabilities as soon as they are discovered, while Snyk Insights helps your organization focus on and prioritize top risks for your organization.



Leveraging application security tools that provide actionable zero-day reporting is crucial for maintaining a strong security posture. These tools help security teams shift their focus from a reactive to a proactive state by immediately highlighting newly discovered, unpatched vulnerabilities. This actionable reporting goes beyond simple alerts by providing developers with the context needed to understand the threat, pinpoint the affected code, and prioritize remediation efforts effectively. Snyk Analytics includes a custom Zero-Day Report fed by the latest zero-day vulnerability intelligence that allows customers to pinpoint their potential areas of exposure and prioritize them for remediation.

In addition, our industry-leading security intelligence stays up-to-date on the latest zero-days and other threats and then uses this intel to power our solutions.

Learn more about our developer-first AppSec solutions to fix security issues throughout the SDLC and avoid becoming a victim of the next zero-day vulnerability.

Prepare for zero-day vulnerabilities with Snyk

Learn More Start Free