Managing vulnerabilities in open source projects

**Overview**

All software, whether open source or not, can contain security vulnerabilities or defects that someone could use to exploit the confidentiality, integrity, or availability of data in a system. This document answers common questions about security vulnerabilities in open source software, offers a brief guide to open source vulnerability management processes, and explains how Red Hat participates in these processes—making open source software more secure for customers.

**Question:** What is a security vulnerability?

**Answer:** A security vulnerability is a defect in computer software or hardware that someone could exploit to affect the confidentiality (privacy), integrity (accuracy), or availability (authorized access) of data in a system. For example, by exploiting a security vulnerability, someone might acquire or view private data, tamper with or change those data, or make those data inaccessible to people who need them.

**Question:** Does open source software contain security vulnerabilities?

**Answer:** Yes. All software, whether open source or not, can contain security vulnerabilities. The way software is licensed, however, can affect how and when its security vulnerabilities get addressed. When an application’s source code is open—that is, transparent and accessible for anyone to examine—more people are able to discover potential vulnerabilities in it. Because people can view the software’s source code, they are also able to determine whether an application contains security vulnerabilities (or “audit” the software) before they begin using it.

Rather than restricting access to source code to prevent discovery of security vulnerabilities (an approach some call “security through obscurity”), open source software projects address vulnerabilities differently: they maintain rigorous standards for transparency and collaborative peer review.

**Question:** How do open source software communities and vendors coordinate and address potential security vulnerabilities?

**Answer:** Members of many open source projects are security experts from prominent vendors in the projects’ wider ecosystems. Larger projects may even have a dedicated group of people who act as the project’s security team. These teams share common communication channels, testing infrastructure, and defect tracking systems. When someone reports a potential software vulnerability through one of these channels, this security team will assess the potential vulnerability and collaborate on a method for addressing it. In their communications, the security team includes the party initially reporting the vulnerability. And as they work to remediate the vulnerability and build a fix, these teams involve the party that initially reported the vulnerability as much as they possibly can.

Most security teams submit incoming reports of potential vulnerabilities to verification and triage systems before assigning specific people to begin working on ways to fix proposed issues. Members of the team review, debate, and perform tests to determine the best solution for correcting the vulnerability. When the patch is ready, the team communicates information about the vulnerability (as well as their plans for issuing patches that fix it) to software vendors and users. Normally, they do this through a project mailing list (or through broader channels, depending on the size and scale of the project). Vendors in the community typically employ product security engineers who staff and monitor these lists.
**Question:** How are security issues in open source applications typically reported?

**Answer:** Many open source projects have developed processes for discovering, fixing, and communicating information about security vulnerabilities. They typically adopt some degree of automation to test for vulnerabilities before releasing software. Many projects coordinate the publication of vulnerability details and the fixes for them with users whose software relies on the affected project.

Processes like these minimize the time users are exposed to security vulnerabilities. Their intent is to give everyone impacted by the vulnerability access to fixes at the same time, so no group is subjected to more risk than others.

Open source projects typically encourage the use of protected channels for reporting security issues. For example, they may use encrypted mail to communicate about security vulnerabilities. Open source projects may also restrict the visibility of security-related issues in their issue-tracking systems, making them accessible only to a core security team. This security team triages incoming reports of security issues and confirms purported vulnerabilities.

Initially, only the core security teams and issue reporters are part of security discussions. Limited visibility ensures that others—including people who might attempt to exploit security vulnerabilities—do not become aware of the vulnerability until a fix is available.

**Answer:** After members of a project security team have identified and confirmed a vulnerability in their open source software, they will use their mailing lists and other communication channels to notify people who depend on their software. Project security team members will grant these communities or commercial vendors an opportunity to review the code that fixes the issue (or "patches the vulnerability") and adjust it to their particular circumstances before they reveal the vulnerability to the broader public.

This act of deliberately withholding information about a known security vulnerability until a fix is ready is called an "embargo." Ideally, an embargo only lasts for a short period of time—long enough to facilitate adequate review of the proposed fixes and notification of stakeholders who need to take action. The longer an embargo lasts, the more likely the vulnerability is to be discovered and possibly exploited before preparations are in place.

When members of a security team reveal the vulnerability (or "lift the embargo"), they typically communicate it through a coordinated response, disclosing specific details about the vulnerability. Security team members frequently publish the vulnerability details in a **CVE (Common Vulnerabilities and Exposures)** document (an industry-standard format). A CVE is a universal identifier of a single, unique vulnerability. The software flaw may exist in various implementations of a package or library—and that flaw may impact different installations differently—but everyone affected by the issue will use the same CVE document to describe the core vulnerability. Centralized communication gives consumers of the software a common understanding of the security scenario. Moreover, a CVE document usually includes a list of vendors who have already issued patches for the security vulnerability.
Question: How can I protect my open source software from security vulnerabilities?

Answer: To remain protected from security vulnerabilities, users should routinely review changelogs and updates from the projects they use. They should also monitor security databases to check for newly discovered issues. Information Sharing Analysis Centers (ISACs) and Community Emergency Response Teams (CERTs) around the globe will also aggregate and communicate known security vulnerabilities (though often with a delay), so organizations maintaining their own open source software should also review these sources. Organizations should ensure system administrators are subscribed to critical projects’ security communications channels, so they are aware of vulnerabilities as soon as those vulnerabilities are reported.

Working with an open source software vendor like Red Hat, which routinely monitors open source security communications and supplies patches for known vulnerabilities (often before those vulnerabilities are widely known), is another way to remain protected. Red Hat, for example, regularly monitors roughly 450,000 open source software packages to protect customers from security vulnerabilities.

Question: How does Red Hat help make open source software more secure?

Answer: Because Red Hat enhances open source software at all stages of the open source software life cycle, we can make open source software more secure in several important ways.

We act as a trusted partner to many upstream open source projects, assisting them with security issues and processes. We also participate in various security teams and lead security initiatives on key industry projects.

Moreover, Red Hat’s Product Security team analyzes and independently verifies every security issue affecting software in any of our products or services. The team reviews packages for known and potential vulnerabilities and assigns every CVE impacting a Red Hat product both a Common Vulnerability Scoring System (CVSS) severity rating and a Red Hat severity rating, which we share with the broader open source ecosystem and industry. To accomplish this, we collaborate with approximately hundreds of open source software communities.

Red Hat customers receive timely, automated updates as part of a Red Hat subscription. These updates contain patches that fix known security vulnerabilities and are frequently available to customers even before particular security vulnerabilities are widely known. We do this even in cases where open source communities have ceased issuing security fixes for their software projects.

About Red Hat

Red Hat is the world’s leading provider of enterprise open source software solutions, using a community-powered approach to deliver reliable and high performing Linux, hybrid cloud, container, and Kubernetes technologies. Red Hat helps customers integrate new and existing IT applications, develop cloud-native applications, standardize on our industry leading operating system, and automate, secure, and manage complex environments. Award winning support, training, and consulting services make Red Hat a trusted adviser to the Fortune 500. As a strategic partner to cloud providers, system integrators, application vendors, customers, and open source communities, Red Hat can help organizations prepare for the digital future.