When the Last Line of Defense is the Weakest

# The Answer Was Already on the Shelf

**How public money built defenses for open source software (way) before the law required them.** *By Jeffrey A. McGuire*

Around 2016, Armijn Hemel took apart a KVM-over-IP switch. The box a sysadmin reaches for when a server stops answering, it opens a path into the machine over the network. This one ran a version of Linux that, by Hemel's estimate, was about 16 years old. The equipment designed to be a data center's last line of defense was the least defended thing in the rack.

Hemel dismantles firmware for a living, so this pattern is familiar. The devices most people never think about (the home router, the cheap camera) tend to ship with already-old software that then runs for years. The owner rarely notices when one gets exploited, because the camera still films and the router still routes. "Would you be okay if a team of burglars set up shop in your home?" he asks. That's roughly what happened with Mirai: malware that, without obvious disruption, used hundreds of thousands of Internet protocol (IP) cameras and similar devices to launch attacks.

Old flaws linger for the same reason: No one is forced to fix them. Philippe Ombredanne, who leads the open source project ScanCode, points to log4j to show "the inertia is huge." A small, ubiquitous piece of Java software, log4j was found in December 2021 to be dangerously vulnerable – and present in a vast number of Internet-facing systems, including banks. More than four years later, a contact at one of the largest cloud providers told Ombredanne that more than 20 percent of its Java customers were still running a flaw that was already years old at discovery.

The scale of this gap is almost invisible to the people relying on the software. Ombredanne likes a thought experiment: Subtract all the open source software from the world and see what stops. The phones, the computer, the networks that connect them. Most cars built in the last few decades and the pumps that would refuel them. Cash registers and the banks behind them. "It would be total chaos," he says.

For years, a small number of people have been building the defenses for this hidden layer of dependency. They did it quietly, on public money, before any law required it. Now a law has arrived.

## A Supply Tree, Not a Supply Chain

The company listed on a typical consumer device box is often little more than a sales office: a support engineer, someone to answer the phone. A step back, the original design manufacturer, or ODM, builds the anonymous white-box hardware that later gets a brand attached; ODMs frequently pick a design from a chipset catalog, Hemel says, the way you might order "that one, in blue." The chipset maker hands the ODM a software development kit (SDK) to drive its silicon, and that SDK is often already at the end of its supported life by the time the finished device reaches a shelf. Hemel once visited a chipset maker at an ODM's request and learned that the chip going into a ready-to-launch product had been discontinued two years earlier.

The Common Vulnerabilities and Exposures (CVE) system was designed decades ago to pin a flaw to a vendor and a product: Microsoft, Adobe, Windows, Acrobat. It was never built for a world where tens of millions of devices share one WiFi chipset and the same aging code. A vulnerability, Ombredanne argues, should be treated as a software attribute and tracked across every device that carries it, rather than as a problem belonging to one vendor. "That's how it should be," he says. "It's not how it is."

This is why Hemel talks about a supply tree rather than a supply chain. One root feeds many near-identical devices wearing different logos. Trace enough Blu-ray recorders back, and they converge on the chipset maker MediaTek. The root of the tree – the chipset and the ODM – is also the root of the problem; the branding companies sit out at the branches, closest to the customer and furthest from the code.

Hemel points to a book, *The Box* by Marc Levinson, about how the standardized shipping container collapsed the cost of moving goods and reshaped global trade. Linux was the embedded software's version of the container, a standard base that let manufacturers stop paying for proprietary real-time operating systems and drove the cost of a capable device toward the floor. Security was the thing that gave way to market pressure: "Fast, cheap, secure," Hemel says. "Pick two."

The result is a strange allocation of risk. The company whose sticker is on the box carries the product's legal liability. But in nearly every case, that company does not know what software is inside – and cannot fix it.

A European law is about to make device safety someone's legal duty. The Cyber Resilience Act, or CRA, requires anything sold with software in it to be built securely and kept patched for years. Ombredanne and Hemel would sharpen that summary: The law rolls out by sector and will prioritize genuine risks. Its first hard deadline, mandatory reporting of exploited vulnerabilities, lands on September 11, 2026; the full set of obligations follows in December 2027. The law reaches toward the root of the tree, even though the liability starts out at the branches.

## The Toolkit's Ready, the Data's in Poor Shape

The CRA will require an inventory of every component in a product, with its licenses and dependencies: a software bill of materials, or SBOM. The tools to produce that list already exist. Reconstructing that list after the fact, when nobody kept the label, is the discipline of software composition analysis – and it requires reliable data.

Ask Ombredanne what his tools do, without a demo, and he reaches for LEGO. ScanCode identifies the origin and license of third-party components inside a piece of software. "You built a LEGO car," he says. "It tells you which bricks you used and what color they are." Its companion, VulnerableCode, adds the security layer. "It tells you this brick here has a crack. Don't use it." Between them, they answer the two questions any company facing the CRA must know: What is in my product, and which parts of it are known to be dangerous?

Tools are the easy half. The hard half is the data they run on. Ombredanne's team keeps finding vulnerability reports for packages that do not exist: a flaw filed against an unreleased version or a package name that appears nowhere anyone actually publishes software. He attributes the flood to AI systems generating plausible-looking reports. Nobody can use software that isn't real, but the fake report lends the fake name credibility; an attacker can then register that name, point to the vulnerability database as proof it exists, and wait for someone to pull it in. Even the real reports arrive faster than anyone can sort them. Ombredanne compares it to an emergency room where a thousand people with paper cuts and one person about to die all sit in the same waiting room, looking alike. And the incentives make it worse. Fame and money in security flow to discovering vulnerabilities, not to fixing them; the Linux kernel's security team holds, by his count, a standing list of more than 500 serious open issues and keeps receiving more reports than fixes.

The work of getting good data is unglamorous and often lonely. To make sense of the security advisories from NGINX, the web server behind a large share of the Internet, Ombredanne's team spent a month decoding a format whose conventions lived mostly in the authors' heads. When he asked the Debian project which license covered its security data, so he could legally reuse it, he was told he was the first person ever to ask.

Data also degrades as it travels. Ombredanne traces one Apache log4j advisory through its journey: Correct at the source, it passed to the MITRE CVE project, then to the US National Vulnerability Database where a format conversion damaged the contents of the version list. GitHub drew its feed from there and damaged it a little more, and GitLab and Google drew theirs from GitHub. The upstream project had it right; a US federal agency, a major open source company and two of the largest technology firms all ended up with a version that was wrong. "It's a game of telephone," Hemel says, and his real objection is the opacity. When the data is open, you can find where it broke; when it is closed, you cannot.

Ombredanne is blunt about why the mess persists. The commercial database and scanning-tool vendors that are the paid competitors to his own tools have little incentive to clean it up, he argues, because the complexity is what justifies their pricing. He points to a US

venture-backed startup selling "CVE-free" container images without contributing its fixes back upstream. To him, that is a kind of enclosure: "You don't put a tax on oxygen," so why require paid access to vulnerability data that's just as vital? In his view, these records are a commons that has to stay open – not merely visible, but licensed for reuse and produced through a process anyone can inspect. That covers three kinds of data: which versions are vulnerable, what license a component carries, and where a component actually came from.

None of this looked like a product anyone would buy. The risk of shipping insecure software was low, and the people who understood the problem were rarely the people holding the budget. Asked whether the market would ever have paid for an open vulnerability database or firmware scanner, Ombredanne does not hesitate. "No," he says." "I don't think so."

Public money provided the answer. Hemel started out of frustration and on his own time. He kept his tools alive with Dutch research and development tax credits and a series of public grants that include support through NGI Zero, the EU-funded programs that fund open source research and development. One product of that work is DeviceCode, an open, searchable catalog assembled by cleaning and unifying enthusiast-sourced facts about real hardware. In a world where vendors and manufacturers keep the contents of our devices obscured, it can list every device shipped with the password "password" and filter the results by ODM and chipset. Though Hemel is the first to admit it is still too raw for a non-technical user, it combines resources with tools like VulnerableCode to track vulnerabilities across clusters of similar devices. Ombredanne's ScanCode has drawn on NGI Zero, as well, on the German Sovereign Tech Agency, and on grants from companies including Bloomberg, Zeiss, Porsche, Mercedes, and Google. He is now building the AboutCode Foundation, a member-funded nonprofit, to keep the work going.

Both men have landed in the same place: CodeSupply. A project funded by the European Commission, it catalogs the software and firmware inside real devices and records each binary's origin, license, health, and security as open data. "More than the tools," Ombredanne says, "the frontier is good data."

## EU CRA: From Threat to Answer

Most of the open source world first heard the CRA as a threat. For the people who had spent years building the required tools, the law's arrival felt like something else. "For me, it was 'finally'," Hemel says.

He casts the law and tooling as two sides of one push. The CRA is the stick; the open, high-quality data of a project like CodeSupply is the carrot, the thing that keeps the compliance overhead as low as it will go. "Though a carrot," he adds, "can also be used as a very short, and probably orange, stick."

Pressed to make the case for the CRA as the answer rather than a fresh burden, Ombredanne lands on one word. "It forces transparency," he says, "or at least more transparency," and calls that a real first step. He is honest about its limits. He and Hemel both think the law would work better organized around software packages rather than finished products. Treat the package as the unit, Hemel says, and security and license information becomes something you simply attach to it, "like decorating a Christmas tree." Ombredanne had hoped the reporting rules might let Europe gather every SBOM into a single census of what open source software is actually in use across the continent; that is not in the plan.

He is just as honest about readiness. The requirement to report an exploited vulnerability inside a tight window is, in his words, close to physically impossible for most companies today. They mostly do not yet know what software their products contain, and the first deadline is set for September 2026. Faced with the choice, many companies will quietly under-report and hope. "The data is shit," he says, "and the tools are not complete."

That is the case for building Europe's own foundation rather than borrowing someone else's. Whoever decides what counts as an exploitable vulnerability decides how much compliance work a company has to do. Ombredanne puts the question plainly: Should a European organization depend on a defunded US federal agency to define its legal obligations? He is not describing a hypothetical. The US National Vulnerability Database, the resource much of the world's security automation quietly relies on, spent 2024 watching its backlog of unanalyzed vulnerabilities climb past 27,000. In April 2026, the institute that runs it conceded it could not keep up and narrowed its full analysis to prioritize US federal and US-critical systems: roughly 15 to 20 percent of incoming vulnerabilities, by independent estimates. A month later, a US government watchdog reported the database could no longer be considered reliable. Europe had no vote in any of the decisions that led there.

The answer was already on the shelf when the law came looking for it: two European efforts. VulnerableCode is the community side, an open database that cross-checks many sources instead of trusting any single one. The European Union Agency for Cybersecurity is the institutional side, standing up a European vulnerability database and moving to take a senior role in the global CVE system. Rather than a European data source for its own sake, "it's about having a correct, open, verifiable, traceable data source," Ombredanne says.

What an ordinary buyer might notice 10 years from now is quieter than the politics and more concrete: a device whose code stays maintained and secure for longer. What changes for the companies shipping that software, in Ombredanne's telling, is a pair of habits that come to seem ordinary: tracking the security of every dependency and paying back into the open source projects they build on, in cash or in kind. Hemel expects new markets to open alongside that development, hardware sold on full traceability and long-term support rather than on price.

Hemel also expects businesses to find angles the legislators never imagined, such as companies spun up to ship a product and dissolved before anyone can hold them to account – with escrow and mandatory pre-market audits as the eventual counter, and at the cost of slower time to market.

"The CRA is not the end of the game," he says. "It is probably just the end of the beginning of the game. We are in for a ride."

Ombredanne's horizon runs further out. Peer review and decentralization are only a step, he says, toward an endgame in which open source projects publish the security and license data for their own code, and no single government's funding decisions can degrade what the rest of the world relies on. An open, independently governed record of what is inside a product and what is wrong with it is not a convenience. It is what digital autonomy looks like in practice: the ability to see clearly, and to fix what you find, without asking anyone's permission.

*This article was made possible by support from NLnet Foundation through Linux New Media's Topic Subsidy Program (https://www.linuxnewmedia.com/Topic_Subsidy).* NLnet Foundation is the Coordinator of the NGI Zero Consortium. NGI Zero programs are made possible with financial support from the European Commission.

*Disclosure: Philippe Ombredanne co-maintains ScanCode, VulnerableCode, and other AboutCode tools listed below and authored the Package-URL (PURL) specification.*

Links for further reading:

- *The Cyber Resilience Act (European Commission)*
- *Evaluation of NIST's Management of the National Vulnerability Database, Report OIG-26-020-I (US Department of Commerce Office of Inspector General)*
- *EU Vulnerability Database (EUVD), ENISA*
- *CodeSupply, the open, federated software-metadata pilot (NLnet, Open Internet Stack)*
- *Software Supply Chain ecosystem supported by NGI Zero*

Links for your toolkit:

A minimum practice for the CRA: Keep a continuous inventory of third-party code inside your products, then track its licenses and vulnerabilities against that inventory, keyed by Package URL. Ombredanne points to a ready-now stack: Dependency-Track or SecObserve to watch the inventory and Syft and Grype to generate and scan a software bill of materials backed by open vulnerability data from OSV, Vulnerability-Lookup, PurlDB, and VulnerableCode.

- *AboutCode*
- *ScanCode Toolkit (GitHub)*
- *ScanCode.io (GitHub)*
- *VulnerableCode (GitHub)*
- *PurlDB (GitHub)*
- *Package URL (PURL) specification (GitHub)*
- *Armijn Hemel's firmware tools, BANG and DeviceCode (GitHub)*
- *Dependency-Track (OWASP)*
- *Syft, SBOM generation (Anchore, GitHub)*
- *Grype, vulnerability scanning (Anchore, GitHub)*
- *OSV, Open Source Vulnerabilities*