CYBER STATECRAFT INITIATIVE
DFRLab

#ACcyber

ISSUE BRIEF
OCTOBER 2023

Driving Software Recalls: Manufacturing Supply Chain Best Practices for Open Source Consumption

BY JEFF WAYMAN AND BRIAN FOX

The **Cyber Statecraft Initiative** works at the nexus of geopolitics and cybersecurity to craft strategies to help shape the conduct of statecraft and to better inform and secure users of technology. This work extends through the competition of state and non-state actors, the security of the internet and computing systems, the safety of operational technology and physical systems, and the communities of cyberspace. The Initiative convenes a diverse network of passionate and knowledgeable contributors, bridging the gap among technical, policy, and user communities.

The mission of the **Digital Forensic Research Lab (DFRLab)** is to identify, expose, and explain disinformation where and when it occurs using open-source research; to promote objective truth as a foundation of government for and by people; to protect democratic institutions and norms from those who would seek to undermine them in the digital engagement space; to create a new model of expertise adapted for impact and real-world results; and to forge digital resilience at a time when humans are more interconnected than at any point in history, by building the world's leading hub of digital forensic analysts tracking events in governance, technology, and security.

EXECUTIVE SUMMARY

In December 2021, the Log4j vulnerability, Log4shell, crippled development teams worldwide. The exploit itself was not what dragged teams down. Instead, most software organizations could not identify whether and where they were using Log4j. This meant developers needed to review entire codebases to determine their exposure and risk. For large enterprises with thousands or tens of thousands of applications, work on new features came to a halt.

Log4shell was another example of software organizations failing to acknowledge or recognize (likely both) that open source software (OSS) is more than just a technological innovation—OSS wholly changed how software products are created. Over more than two decades, OSS catalyzed an already growing movement towards componentized software development—where applications are developed in parts by different internal and external teams. In many ways, OSS transformed the industry into something that more closely mirrors traditional manufacturing.

While there is no 1:1 comparison between software development and other forms of manufacturing, there are still many similarities that provide a learning opportunity. Specifically, by looking at automotive manufacturing, there are modern supply chain management best practices capable of improving OSS consumption and software supply chain security. These same mechanisms can also improve the processes software manufacturers use to disclose the presence of vulnerabilities to their customers.

It is the latter point that is most critical. Like expectations set for any other manufacturer, customers expect software manufacturers to follow a standard of care to ensure their products are safe and secure by design. More importantly, if there is a defect in a product, customers expect a manufacturer to communicate and remediate the defect.

Traditionally, this process of notification and remediation is referred to as a recall. Like comparing software and automotive manufacturing, a recall process for software cannot be applied exactly. Yet, there are similarities. Critical elements of recall processes could provide a mechanism to hold software manufacturers accountable for the parts they use and for communication of critical vulnerabilities (defects) to their customers. However, many manufacturers do not, or cannot, track the OSS used in their software products. Worse yet, in many cases, they lack awareness of critical vulnerabilities in the software products they manufacture.

This paper aims to demonstrate how principles from modern automotive manufacturing, specifically those from W. Edwards Deming, a leader in supply chain and management theory, can be applied to improve OSS consumption and supply chain security. With these processes in place, software manufacturers can minimize the impact of vulnerable OSS and communicate to customers when those defects are encountered. To implement these improvements, policy from the federal government will need to provide further guidance, direction, and accountability.

For software manufacturers, this means:

- Building security into software products by design.
- Consuming only high-value OSS components and projects.
- Continuously tracking, monitoring, and improving OSS consumption.

For policymakers, this means:

- Holding software manufacturers responsible and accountable via a national standard of care.
- Requiring software manufacturers to demonstrate their approach to vetting OSS used in their products.
- Driving software manufacturers to continuously track, monitor, and improve OSS consumption.

INTRODUCTION

On December 10, 2021, around three weeks before much of the world logged off for winter and end-of-year holiday festivities, arguably the worst software vulnerability ever discovered, Log4shell,[1] was publicly disclosed. And this is not something said lightly. Following heavy-hitting vulnerabilities like Heartbleed and Shellshock, Log4shell had an unprecedented impact.[2] The vulnerability affected Log4j, a ubiquitous open source logging framework used to track information and errors in computer systems.[3]

Logging tools provide critical functionality to software organizations today, helping enterprises investigate and determine causes of unexpected operations of everything from websites to applications on your phone. For example, if a server suddenly shuts down, those logs help pinpoint the root cause. Log4j, which the Apache Software Foundation manages, is used in almost every Java application,[4] especially at the enterprise level.[5] Log4shell was critical both because it was easy to exploit and due to its potential widespread impact, which included servers providing critical access to secured networks and sensitive data at the private, commercial, and national levels.

The US Cybersecurity and Infrastructure Security Agency (CISA) has cited the scale of the Log4shell vulnerability across much of their published best practices. In July 2022, the first report from the Cyber Safety Review Board (CSRB) provided updates on lessons learned from Log4shell as well.[6] And then in March 2023, the National Cybersecurity Strategy stressed the importance of open source software (OSS) security and its impact on supply chains. Unfortunately, with the historical absence of meaningful cybersecurity regulatory oversight, organizations and individuals must often voluntarily adopt these best practices and recommendations, especially in engagements outside government activities.[7] This gap is punctuated by evidence indicating this is not happening across the board.

In 2022, a year after the disclosure of Log4shell, a study of current Log4j downloads indicated that as much as 30 percent of users were still using vulnerable versions.[8] Some of these cases were potentially due to hubris or lack of care. However, the more likely cause of continued downloads of vulnerable versions of Log4j was an organization's lack of visibility into the OSS they consume. Without this insight,

1 "CVE-2021-44228," NIST NVD, December 10, 2021, https://nvd.nist.gov/vuln/detail/CVE-2021-44228.

2 "The Heartbleed Bug," Heartbleed, June 3, 2020, https://heartbleed.com/; "GNU Bourne-Again Shell (Bash) 'Shellshock' Vulnerability," CISA, September 30, 2016, https://www.cisa.gov/news-events/alerts/2014/09/25/gnu-bourne-again-shell-bash-shellshock-vulnerability-cve-2014-6271.

3 "Log4j – Apache Log4jTM 2," The Apache Software Foundation, May 2, 2023, https://logging.apache.org/log4j/2.x/.

4 This includes notable systems and products like Amazon Web Services, Cloudflare, and iCloud, among others. Due to the lack of disclosure requirements the exact impact of the vulnerability is impossible know.

5 Liam Tung, "US warns Log4j flaw puts hundreds of millions of devices at risk," *ZDNET*, December 14, 2021, https://www.zdnet.com/article/log4j-flaw-puts-hundreds-of-millions-of-devices-at-risk-says-us-cybersecurity-agency/.

6 Cyber Safety Review Board, "Review of the December 2021 Log4j Event," July 11, 2022, https://www.cisa.gov/sites/default/files/publications/CSRB-Report-on-Log4-July-11-2022_508.pdf.

7 "FACT SHEET: Biden-Harris Administration Announces National Cybersecurity Strategy," The White House, March 2, 2023, https://www.whitehouse.gov/briefing-room/statements-releases/2023/03/02/fact-sheet-biden-harris-administration-announces-national-cybersecurity-strategy/.

8 "Open Source Supply, Demand, and Security," Sonatype, https://www.sonatype.com/state-of-the-software-supply-chain/open-source-supply-demand-security.

organizations cannot effectively respond to vulnerabilities, including communication of the presence and effect of those vulnerabilities on users (customers) of their software.

There is a general expectation that products like food, vehicles, and other goods should be inherently safe. In cases where products include components known to be harmful or defective, manufacturers have a responsibility to disclose and remediate that risk through a recall process. When this is impossible, manufacturers are usually obligated by regulatory policy to warn their customers of the potential danger of defective products. Yet, software products are not held to the same standard; this must change.

When defects are present, like their peers, software manufacturers are responsible for communicating potential risks to their users and guiding them through remediation options. Fortunately, addressing the awareness of OSS consumption and improving communication related to OSS vulnerability disclosure does not require every aspect of a typical recall process deployed in manufacturing. However, achieving those improvements does necessitate software manufacturers track and monitor all the OSS they consume and incorporate into their software products.

Starting with an analysis of the Log4j vulnerability and the corresponding response by software manufacturers, this paper aims to provide a better understanding of OSS consumption, the role OSS plays in the modern software supply chain, and relevant parallels to traditional manufacturing, specifically in the automotive sector. This comparison provides an opportunity to borrow essential mechanisms tested across many years against similar challenges. Looking specifically at automotive manufacturing also provides an opportunity to isolate the best and most relevant examples, especially those pioneered by W. Edwards Deming, a pivotal figure in modern supply chain management. By building awareness of OSS consumption, software manufacturers can improve their ability to effectively respond to issues like Log4shell and facilitate risk communication through existing coordinated vulnerability disclosure (CVD) practices, such as publishing advisories and notifications, that closely resemble recall processes in manufacturing. While disclosure and communication are critically important, this paper's primary intent is to assert software manufacturers' responsibility to continuously track, monitor, and improve their consumption of OSS at an organizational level.

THE SEVERITY OF LOG4SHELL

Log4shell (CVE-2021-44228)[9] earned the highest Common Vulnerability Scoring System (CVSS) score, level 10 (critical), in its official Common Vulnerability Enumeration (CVE) disclosure.[10,11] For perspective, an independent database of CVSS scores shows just 4 percent of all CVEs ever recorded (over 200,000 through twenty-three years of reporting) received a score of ten, which is typically limited to high-impact vulnerabilities that are also easy to exploit.[12]

In the case of Log4shell, the vulnerability allowed remote code execution—the ability for bad actors to remotely make changes to, run software on, and take control of a system. While this type of exploit is terrible in any situation, what made Log4shell so potentially dangerous was its ubiquity within the Java ecosystem. In an article from *Wired* magazine shortly after the official disclosure, Log4shell was characterized as something that would "haunt the Internet for years."[13] Echoing that sentiment, just four days after the Log4shell disclosure, CISA Director Jen Easterly briefed industry leaders on the situation, saying, "[the exploit] is one of the most serious I've seen in my entire career, if not the most serious."[14] Jay Gazlay of CISA's Vulnerability Management Office followed Easterly's comments, stating, "Hundreds of millions of devices are likely to be affected." However, that number is likely low, given estimates of the breadth of affected companies and projections from experts that the vulnerability will persist for years to come.[15]

By January 2022, there were already multiple reported examples of bad actors exploiting the Log4shell vulnerability.[16] By September of the same year, the US government published an advisory confirming that the Federal Civilian Executive Branch (FCEB) had been compromised.[17] Considering the Log4shell vulnerability was present in versions of Log4j since 2013, there is a high likelihood that attacks took place for some time before the official disclosure.

9 "CVE-2021-44228 Detail," December 10, 2020, https://nvd.nist.gov/vuln/detail/CVE-2021-44228.

10 While CVSS scores provide a common framework to compare cybersecurity vulnerabilities, it should be noted that scores are imperfect and can be misleading. For more, read Jacques Chester's "A closer look at CVSS scores."

11 "Log4j – Apache Log4j Security Vulnerabilities," The Apache Software Foundation, https://logging.apache.org/log4j/2.x/security.html#fixed-in-log4j-2-15-0-java-8.

12 "CVE Security Vulnerability Database. Security Vulnerabilities, Exploits, References and More," CVEdetails, https://www.cvedetails.com/index.php.

13 Lily Hay Newman, "The Log4j Vulnerability Will Haunt the Internet for Years," *Wired*, December 14, 2021, https://www.wired.com/story/log4j-log4shell/.

14 Tim Starks, "CISA Warns 'Most Serious' Log4j Vulnerability Likely to Affect Hundreds of Millions of Devices," *CyberScoop*, December 13, 2021, https://cyberscoop.com/log4j-cisa-easterly-most-serious/.

15 Jonathan Greig, "Log j Update: Experts Say Log4shell Exploits Will Persist for 'Months If Not Years,' *ZDNET*, December 13, 2021, https://www.zdnet.com/article/log4j-update-experts-say-log4shell-exploits-will-persist-for-months-if-not-years/.

16 Bill Toulas, "State Hackers Use New PowerShell Backdoor in Log4j Attacks," *BleepingComputer*, January 11, 2022, https://www.bleepingcomputer.com/news/security/state-hackers-use-new-powershell-backdoor-in-log4j-attacks/.

17 "Iranian Government-Sponsored APT Actors Compromise Federal Network, Deploy Crypto Miner, Credential Harvester," CISA, November 25, 2022, https://www.cisa.gov/news-events/cybersecurity-advisories/aa22-320a#:~:text=fbi.gov.-,Mitigations,-CISA%20and%20FBI.

Unfortunately, exploitation of a vulnerability does not necessarily mean software manufacturers will pay attention. CISA's Cybersecurity Strategic Plan FY 2024-2026 highlights that "most intrusions today are perpetrated using known vulnerabilities or exploiting weak security controls."[18] A telemetry analysis from Tenable, a cybersecurity risk firm, found that 72 percent of organizations were still vulnerable to Log4shell as of October 2022.[19] To understand how that is possible, when this article was written, a review of Maven Central, the largest repository of open source Java components, showed that 23 percent of new downloads each week represented versions vulnerable to Log4shell.[20] This equates to hundreds of thousands of vulnerable versions of Log4j entering software supply chains every month.

However, the negativity surrounding the Log4shell vulnerability is only part of the story. The spotlight on Log4j demonstrates OSS's tremendous impact on modern software development. If software manufacturers are unaware of the vulnerability of critical OSS like Log4j, what about all the other OSS they consume?

THE PROBLEM IS NOT OPEN SOURCE SOFTWARE

Today, almost 90 percent of modern applications are composed of OSS, including components like the Log4j logging framework.[21] On average, about 11 percent of those OSS components have known vulnerabilities. While the extensive use of OSS has reduced the cost of research and development, catalyzed incredible leaps in innovation, and drastically decreased the time it takes to deliver critical business functionality, it has also created a security conundrum.

Unlike other third-party software, in most cases, OSS is not "supplied;" that is, projects or individual developers that maintain the software do not act as suppliers. As described in Chinmayi Sharma's "Tragedy of the Digital Commons," OSS is a "public good" or a natural resource freely available to anyone.[22] As a result, its creators cannot know how their work will be used or, in many cases, how software manufacturers may modify it to fit their customers' needs. Instead, it is the responsibility of the software manufacturer to ensure that any final product delivered to a customer that uses OSS is free of defects (i.e. vulnerabilities). This relationship differs from traditional supplier engagement and procurement processes used to offset research and development costs. This difference presents considerable potential for risk.

A componentized approach to software development is not new. Many organizations still outsource the development of specific frameworks and other elements of an application to third parties. However, these engagements use standardized procurement and review processes that ensure products match technical specifications and other contractual requirements. In contrast, for many organizations, the consumption of OSS has no equivalent process.

This lack of process is both a blessing and a curse. Development teams can use and modify whatever they find among the vast amount of available OSS. That freedom allows them to completely sidestep the procurement process used when working directly with third parties. This circumvention often is not done intentionally. In most cases, development teams simply do not associate OSS consumption with procurement. This lack of overhead can create short-term benefits for innovation. However, knowing *where* OSS is used is just as crucial as *what* OSS is consumed. Without common standards, software manufacturers often do not track their OSS consumption, making it extremely difficult to monitor and identify defects.

By ignoring OSS consumption, development teams expose the organization to increased risk, especially when vulnerabilities are discovered. This extends well beyond Log4shell: the same report that identified continued downloads of vulnerable versions of Log4j also found that 96 percent of OSS downloads with a vulnerability have non-vulnerable updates that are available.[23] The lack of visibility into the consumption of OSS means software manufacturers often miss these fixes and patches that would reduce and, in most cases, entirely remove the risk associated with vulnerabilities in previous versions.

THE IMPACT OF VULNERABILITIES

The term "vulnerabilities" sounds frightening; sometimes, they can have that potential. However, vulnerabilities generally are not an injection of malicious code but rather an inadvertent weakness in the code itself.[24] In many ways, vulnerabilities can be as simple as a typo in any written content. This broad definition is true of all code, open source or proprietary. The nature of a vulnerability changes when it can be exploited. In other words, when those bugs allow bad actors to access private systems, the vulnerability represents risk.

18 "CISA Cybersecurity Strategic Plan FY2024-2026," CISA, August 6, 2023, https://www.cisa.gov/sites/default/files/2023-08/FY2024-2026_Cybersecurity_Strategic_Plan.pdf.

19 "Tenable Research Finds 72% of Organizations Remain Vulnerable to 'Nightmare' Log4j Vulnerability," Tenable, November 30, 2022, https://www.tenable.com/press-releases/tenable-research-finds-72-of-organizations-remain-vulnerable-to-nightmare-log4j.

20 "Log4j exploit updates," Sonatype, https://www.sonatype.com/resources/log4j-vulnerability-resource-center.

21 "2020 State of the Software Supply Chain," Sonatype, https://www.sonatype.com/hubfs/SSC/SON_SSSC-Report-2020_sept23.pdf.

22 Chinmayi Sharma, "Tragedy of the Digital Commons," *North Carolina Law Review* 101 (2023), 1129, https://doi.org/10.2139/ssrn.4245266.

23 "8th Annual State of the Software Supply Chain Report," Sonatype, 2021. https://www.sonatype.com/state-of-the-software-supply-chain/introduction

24 "Vulnerability – Glossary," National Institute of Standards and Technology, https://csrc.nist.gov/glossary/term/vulnerability.

The presence of risk does not mean a vulnerability is inherently critical to every organization and every software product. Depending on the context, vulnerabilities may present little to no risk, even in the case of a critical OSS component like Log4j. However, it is impossible to understand and address a risk without a clear understanding of OSS consumption, implementation, and configuration. This lack of understanding directly impedes software manufacturers' ability to respond to critical issues effectively.

Once again, using the Log4j vulnerability as an example, much of the time development teams spent addressing Log4shell was not focused on implementing fixes and applying patches to Log4j. Instead, the lion's share of initial investment was spent trying to fully understand their use of Log4j in the first place. So, before the arduous technical work could begin, teams first needed to figure out their exact version of Log4j and where the vulnerable versions existed across their portfolio of software products.[25] Again, all this must happen before any fix or patch can be applied. This can quickly become an impossible task at scale for large organizations with a complex code base and tens of thousands of applications.

Regardless of organizational size, the reactive nature of a software manufacturer's response to vulnerabilities is the best example of the current weaknesses in OSS consumption management. Yet, that weakness is almost entirely avoidable. If software organizations track what OSS they consume, where that OSS is used, and then monitor OSS for defects and other quality parameters, their response can be far more proactive. In many cases, issues can be addressed long before a product ships. Even when that is not possible, the ability to triage and prioritize remediation efforts avoids chaos when a vulnerability is discovered, allowing teams to take direct control of the response and tackle defects strategically.

The inability of many development teams to effectively respond to Log4shell should be a call for software manufacturers to change. At the center of this change is the acceptance and adoption of processes that acknowledge OSS is not simply a way to bypass traditional procurement. Instead, OSS must be a critical consideration in managing a software supply chain. Achieving a paradigm shift like this requires tested principles and mechanisms. Luckily, there are a plethora of modern supply chain management best practices that can be borrowed from other manufacturing industries, especially automotive manufacturing.

THE ADVANTAGE OF MODERN SUPPLY CHAIN MANAGEMENT PRINCIPLES

While nuances exist, the intent of this paper is not to draw a direct line between software development and automotive manufacturing. Instead, it is to compare processes in both industries, especially related to supply chain best practices. This is also not to say that manufacturing has everything figured out. Even in recent history, there have been low points, such as the combined failure of General Motors and the National Highway and Transportation Safety Administration (NHTSA) to recall faulty ignition switches in the Chevy Cobalt.[26] However, despite these setbacks and continued opportunities to improve, automobile manufacturers have developed efficient and effective processes for identifying and communicating defective products. Through targeted notifications and safety recalls, automotive manufacturers collectively communicate defects for millions of vehicles each year.[27]

In many cases, recalls are related to discovering and communicating severe safety issues that could cause serious injury and, in some cases, death. In this way, the volume of recalls represents drastic improvements to consumer safety.[28] However, a common misconception is that recalls are a way to pull a defective product back. While this works in some cases, for example, if a vehicle has not yet been sold, most recalls affect vehicles already on the road. This means the manufacturer must be able to identify not only defective parts but also the location of the affected vehicles. The critical point here is that recalls would be impossible without the ability to track parts in a vehicle throughout the supply chain and up to final assembly. Put another way, this ability to track and monitor parts means that when a defect is identified, the manufacturer can target their communication and any remediation steps to the affected consumers.

Of course, tracking and responding to defects is only a part of modern supply chain management. Manufacturers must also work to minimize defects, and this is where modern supply chain theory provides the most relevant and helpful guidance for software supply chains. Specifically, today's software manufacturers should look to the work of W. Edwards Deming, who was responsible for helping rebuild automotive manufacturing in post-World War II Japan and was highly influential in the global automotive market. Most notably, Deming focused on improving supply chain practices and, more importantly, ensuring greater control over quality and security.[29]

25 "(ISC)^2 Pulse Survey: Log4j Remediation Exposes Real-World Toll of the Cybersecurity Workforce Gap," "(ISC)^2, February 22, 2022, https://blog.isc2.org/isc2_blog/2022/02/log4j-remediation-exposes-cybersecurity-workforce-gap.html.

26 Jerry Hirsch, "NHTSA Launches Probe into Cobalt Recall; GM Issues Another Apology," Los Angeles Times, February 27, 2014 https://www.latimes.com/business/autos/la-fi-hy-nhtsa-gm-cobalt-recall-probe-20140227-story.html.

27 Jim Gorzelany, "Automakers with the Most and Fewest Recalls in 2022," *Forbes*, January 2, 2023 https://www.forbes.com/sites/jimgorzelany/2022/12/30/automakers-with-the-most-and-fewest-recalls-in-2022/?sh=441e13327cb9.

28 "Vehicle Safety Recalls Week," NHTSA, https://www.nhtsa.gov/recalls/vehicle-safety-recalls-week#:~:text=Every%20vehicle%20recall%20is%20serious,any%20unrepaired%20recalls%20fixed%20immediately.

29 The W. Edwards Deming Institute, https://deming.org/.

Deming insisted that manufacturers source the best parts from the best suppliers, which was at the heart of his strategy. As part of his recommendations, he put together a fourteen-point approach to quality management.[30] Many of these ideas are now accepted concepts in manufacturing, like the Andon principle, which states that any worker should be able to immediately stop production to prevent defects and further quality issues down the line. While the complete set of fourteen principles dives deeper into management philosophy and is outside the scope of this paper, for software supply chains and improvements to OSS consumption, three are critical:

- **Principle 3:** Cease dependence on inspection to achieve quality.[31] In this principle, Deming suggests manufacturers "shift left." By moving inspection earlier in the production processes, defects are found when changes are much easier to make. Inspection of the final product should still happen but should not be the only or first inspection point.

- **Principle 4:** Move toward a single supplier for any one item on a long-term relationship of loyalty and trust.[32] In this principle, Deming suggests that complexity is introduced by utilizing multiple suppliers for the same part. By utilizing the single, best supplier and building a relationship with them, when defects enter the supply chain, you only need to focus on reaching a resolution with a single supplier versus tackling issues from several suppliers simultaneously.

- **Principle 5:** Constantly improve production systems to improve quality and efficiency, and thus constantly decrease costs.[33] In this principle, Deming aligns with the philosophy that you cannot improve what you do not monitor, and you cannot monitor what you do not track.

It is important to consider that quality can be highly subjective, establishing a singular definition of high-quality OSS is unnecessary. Rather, Deming's principles offer an approach for software manufacturers to develop better processes for the consumption of OSS, which will enhance its quality in the long term. Here is how translating Deming's guidance to software supply chains looks:

- **Principle 1: Build security into software products by design.**[34] Like physical products, software manufacturers should be responsible for ensuring their products are safe and secure. Within the context of liability, this responsibility is often associated with a duty of care. In addition to a duty of care, a manufacturer's level of responsibility to ensure products are safe and secure is a standard of care. To align with the National Cybersecurity Strategy and, in turn, meet a reasonable duty and standard of care, security needs to be a critical part of software manufacturing from the start.[35] For example, assessing the security of OSS in a product only after it is released is too late. Instead, software manufacturers must take an active role in their consumption of OSS at every stage of the Software Development Life Cycle (SDLC).

- **Principle 2: Use only the best, actively supported OSS components and build relationships with those projects and developers.** Selecting the best OSS means evaluating it against criteria like known vulnerabilities, age, and average remediation/update times, among others. When an OSS component meets those standards, manufacturers should utilize it exclusively to avoid duplication and reduce their overall attack surface (risk). Next, select stable, supported versions of OSS and vet projects to ensure they utilize recommended processes and best practices.[36] Finally, build partnerships with high-quality open source projects and invest back into those projects to accelerate innovation upstream and reduce future costs downstream.

- **Principle 3: Continuously track, monitor, and improve the security of OSS that is being consumed.** Manufacturers should understand how and where they consume OSS spanning the entire SDLC to reduce their risk related to known vulnerabilities. Software manufacturers should also establish criteria and develop organizational policies to improve the consumption of OSS. While efforts may start small, research indicates[37] a combination of modern tooling and best practices provide scalable and organization-wide approaches that can be applied across all teams and products without increasing costs or reducing productivity.

30 "Dr. Deming's 14 Points," The W. Edwards Deming Institute, https://deming.org/explore/fourteen-points/.

31 "Inspection Is Too Late. The Quality, Good or Bad, Is Already in the Product," The W. Edwards Deming Institute, November 8, 2012 https://deming.org/inspection-is-too-late-the-quality-good-or-bad-is-already-in-the-product/.

32 "The Importance of Working with Suppliers over the Long Term," May 18, 2015, The W. Edwards Deming Institute https://deming.org/the-importance-of-working-with-suppliers-over-the-long-term/.

33 "Haircuts and Continuous Improvement," The W. Edwards Deming Institute, July 31, 2017, https://deming.org/haircuts-and-continuous-improvement/.

34 "Security-By-Design and -Default," CISA, June 12, 2023, https://www.cisa.gov/resources-tools/resources/secure-by-design-and-default.

35 "Shifting the Balance of Cybersecurity Risk: Principles and Approaches for Security-By-Design and -Default," CISA, April 13, 2023 https://www.cisa.gov/sites/default/files/2023-04/principles_approaches_for_security-by-design-default_508_0.pdf.

36 "OpenSSF Scorecard - Security Health Metrics for Open Source," GitHub, https://github.com/ossf/scorecard.

37 "2020 State of the Software Supply Chain: Chapter 4 - How High Performance Teams Manage Open Source Software Supply Chains," Sonatype, September 23, 2020, https://www.sonatype.com/hubfs/SSC/SON_SSSC-Report-2020_sept23.pdf.

The three principles discussed above should define every software manufacturer's core strategy for OSS consumption and guide their approach to software supply chain security. At the same time, it is worth a word of caution that an over-correction can occur. The removal of all vulnerabilities is not necessary—if such a state is even achievable. As mentioned previously, exploitability and vulnerability mean different things. Many vulnerabilities have little potential for harm.[38] However, when a critical vulnerability does exist, once again, the most applicable lessons for software manufacturers come from a set of mechanisms used by automotive manufacturers to identify and respond to defects.

THE FIRST GOAL OF A RECALL

An automotive recall is usually, although not always, conducted in partnership with the National Highway Transportation and Safety Administration (NHTSA), which, among other activities, investigates defects in automotive products.[39] At the end of an investigation, the NHTSA provides a non-binding recommendation to the manufacturer on how to recall the product.[40] Though automotive manufacturers can go against the recommendation, the NHTSA can seek legal action to ensure vehicle safety standards are upheld. However, any automotive manufacturer will likely say that their first goal for recalls is to avoid them altogether.

While a recall represents improved safety, recalls also represent significant expenses. For example, one consulting firm—AlixPartners—found that 2016, recalls had cost the automotive industry $22.1 billion.[41] Adding a bit more detail, Forbes cited that the average per-vehicle cost of a recall was about $500 over the last 10 years.[42] However, the cost of some recalls is much higher—for example, 2021's Hyundai recall of 82,000 vehicles, which came in at $11,000 per vehicle, sets a new benchmar.[43] Like any business, automotive manufacturers focus on minimizing costs. And in the case of a critical defect requiring a recall, costs can increase exponentially. So, to better mitigate and minimize costs due to defects, many automotive manufacturers utilize processes and best practices related to supply chain management like Deming's principles. This approach allows automotive manufacturers to proactively address defects in production and respond to customers quickly, efficiently, and effectively.

For example, in 2020, Toyota encountered a potential coolant leak defect caused by a faulty water flow meter used to manufacture their engines.[44] In this instance, the supplier identified the issue with the water meter but found no evidence this had created a defect in the engines themselves (it is important to note that Toyota supplies its own engines). Because the supplier "found no abnormalities," it continued to ship the engines for final assembly at a Toyota manufacturing plant. However, according to Toyota's investigation, a coolant leak defect was detected in vehicles awaiting delivery to dealerships, as well as a small number already in dealer inventory. Luckily, Toyota could use serial numbers from the defective engines to trace the leaks back to engines manufactured using the defective water flow meter over a three-month period in 2019. With that information, Toyota conducted an official recall, communicating with dealerships and customers to coordinate inspections and repairs.

For a better understanding of the scale of both tracking the issue and recalling the potentially impacted vehicles, consider that Toyota's engine manufacturing plant in Kentucky (their supplier) produces approximately 600,000 engines a year,[45] and in 2019, Toyota sold over two million vehicles[46] in the United States. To account for all defective parts, Toyota recalled just over 44,000 vehicles. However, the silver lining to this story is that the final number of vehicles impacted by the defect was minimal: only 250, or about 0.05 percent, of those included in the recall.

This example demonstrates the application of all three of Deming's principles. First, inspection was built into the manufacturing process by design and occurred before final assembly. And it's important to note, even with those inspections in place, some defects don't become apparent until they've reached production. Next, the example shows the importance of a strong relationship with suppliers, which made it easier to pinpoint the cause of the engine defect once they appeared after final assembly. In this scenario, the supplier's adherence to supply chain best practices was as important as the manufacturer's. Finally, because Toyota tracks and monitors its parts and vehicles, it was possible to use the serial numbers from engines manufactured with the faulty water flow meter and identify only those vehicles with the potential to leak coolant. Toyota then uses all the data from this investigation in its continuous improvement process for manufacturing.

38 "Do all vulnerabilities really matter?," Red Hat, November 4, 2022, https://www.redhat.com/en/blog/do-all-vulnerabilities-really-matter

39 National Highway Traffic Safety Administration, https://www.nhtsa.gov/.

40 "Safety Issues and Recalls," NHTSA, https://www.nhtsa.gov/recalls.

41 Michael Held, Alexandre Marian, and Jason Reaves, "The auto industry's growing recall problem—and how to fix it," Alix Partners, January 2018, https://www.alixpartners.com/media/14438/ap_auto_industry_recall_problem_jan_2018.pdf.

42 Steve Tengler, "Auto Recalls Way Down in 2023 And Mercedes Knows Why," Forbes, June 28, 2023. https://www.forbes.com/sites/stevetengler/2023/06/28/auto-recalls-way-down-in-2023-and-mercedes-knows-why/?sh=398df6e06795.

43 "Hyundai's recals 82,000 electric cars is one of the most expensive in history," (sp) CNN Business, February 26, 2021, https://www.kktv.com/2021/02/26/hyundais-recals-82000-electric-cars-is-one-of-the-most-expensive-in-history/.

44 "Toyota and Lexus Recall Cars to Replace Engineers," February 14, 2020, https://www.consumerreports.org/car-recalls-defects/toyota-lexus-recall-replace-engine-avalon-camry-rav4-es/; Toyota NHTSA Defect Information Report" February 6, 2020, https://static.nhtsa.gov/odi/rcl/2020/RMISC-20V064-0396.pdf.

45 "Toyota Motor Manufacturing, Kentucky (TMMK)," https://pressroom.toyota.com/facility/toyota-motor-manufacturing-kentucky-tmmk/.

46 "Toyota Motor North America Reports December 2019, Year-End Sales," January 3, 2020, https://pressroom.toyota.com/toyota-motor-north-america-reports-december-2019-year-end-sales/.

A RECALL FOR SOFTWARE MANUFACTURERS ALREADY EXISTS

At first glance, the ability to recall software seems absurd. However, requiring a customer to physically return a software product is too literal an interpretation. Instead, it's better to consider that software manufacturers share a similar goal to automotive manufacturers: to produce a traceable record of defects and vulnerabilities in their products to reduce costs and respond more quickly, effectively, and efficiently. Thus, the lesson from the previous example for software manufacturers is that a recall process like Toyota's and other automotive manufacturers' is already possible. In fact, the best software manufacturers follow a standard process commonly referred to as a Coordinated Vulnerability Disclosure (CVD).[47]

A CVD process is a collaborative approach that typically brings together cybersecurity researchers and software manufacturers to address critical vulnerabilities and provide communication to customers when necessary. To manage the relationship between the members of this group, most software manufacturers publish a vulnerability disclosure policy with criteria and guidance for reporting vulnerabilities, including estimated timelines for remediation and mitigation. At its core, CVD provides a way for software manufacturers to communicate with customers and improve the overall quality and safety of their products. Like the process described in tracing the root cause of Toyota's defective engines, CVDs are most typically initiated by an external identification of an exploitable vulnerability—a defect. In the case of OSS, this process is already happening and is a recommended best practice in the most widely used projects. To understand how this is already in place today, once again, Log4shell provides a valuable point of analysis.

The identification and announcement of the Log4shell vulnerability was part of a standard CVD process and followed many of the same steps as a product recall. In the case of Log4j, the CVD quickly made the headlines worldwide. Log4j would be hard to miss even for manufacturers not tracking or monitoring their OSS consumption. However, the panic that followed did not stem from the potential severity and exploitability of Log4shell alone. While those aspects were important, an even more fundamental issue was that software manufacturers were unaware of where Log4j was used in their applications or if it was used at all. Unlike Toyota engines, most software manufacturers have no serial number equivalent to connect OSS components like Log4j with impacted products. This gap left software manufacturers only one option: look through every application to find a Log4j dependency, often resorting to scanning the disks of production servers. For organizations with tens of thousands of applications, this is the equivalent of Toyota recalling every vehicle they have ever sold to determine which were affected. Not only would this be unacceptable, but the catastrophic cost would burden Toyota for years.

Defects resulting in a recall, or CVD in the case of software manufacturing, test the strength and security of a supply chain. However, best practices and processes alone are not enough. While Deming's principles created measurable improvements for automotive manufacturing, another element is at play in the contrasts between Toyota and software manufacturers: responsibility. Software manufacturers must take responsibility for the security of their software from the start. By design,[48] they must evaluate their suppliers, whether OSS or commercial, and, most importantly, they must continuously track, monitor, and improve their consumption of OSS across all their products and at every stage of the SDLC. Achieving this goal at scale requires a combination of data, processes, best practices, and modern tooling. But most importantly, commitment to the responsibility to deliver safe, secure products.

With these pieces in place, software manufacturers can meet the expectations and standards of their peers in other industries. Software manufacturers will not need to spend months determining which products are affected when the next critical vulnerability, like Log4shell, is identified. Instead, quick and efficient identification will support software manufacturers' ability to utilize disclosure mechanisms like CVDs and proactively communicate mitigation and remediation steps with their customers. While this is not equivalent to removing products from shelves through physical recalls, better communication can still drive reduced risk for customers in the same manner. Further, with the improved consumption of OSS and attention to the guidance outlined throughout this paper, software manufacturers can work to avoid vulnerabilities in the first place.

RECOMMENDATIONS FOR SOFTWARE MANUFACTURERS AND POLICYMAKERS

Imagine that the next critical OSS vulnerability is identified. Could software manufacturers determine which applications in their portfolio are at risk? Could they determine, based on context, if the vulnerability is exploitable? Could they ensure that future downloads are of the non-vulnerable version? How would (or could) they disclose that information to customers? Based on the available data, the most likely answer is no, or at least not without great difficulty.[49]

Many months have passed since Log4shell, yet teams continue to be affected. As of the writing of this paper, vulnerable versions of Log4j still constitute one-third of all Log4j

47 "Coordinated Vulnerability Disclosure Process," CISA, https://www.cisa.gov/coordinated-vulnerability-disclosure-process

48 "CISA Director Easterly Remarks at Carnegie Mellon University," CISA, February 27, 2023, https://www.cisa.gov/cisa-director-easterly-remarks-carnegie-mellon-university

49 "8th Annual State of the Software Supply Chain Report."

downloads. The Log4shell vulnerability has been described as "endemic"[50] and may never go away. Looking beyond Log4j, almost all downloads (96 percent) of vulnerable open source components have a non-vulnerable version available. The logical conclusion is that software manufacturers are unaware of, uninterested in, or perhaps worse, incapable of seriously evaluating their OSS consumption.

The accepted paradigm of inaction and ignorance regarding OSS consumption and software supply chain security is beginning to change. The latest National Cybersecurity Strategy, along with new requirements for government contractors and vendors, is just the first step. Policy and regulations will be revised with even more stringent criteria.[51]

Software manufacturers that follow modern supply chain management best practices and principles described in this paper have an opportunity to address liability concerns and protect their customers from risks associated with unmanaged OSS consumption.[52] Moreover, when critical vulnerabilities occur, software manufacturers can provide effective communication to guide their customers through a recall-like disclosure process that addresses steps for mitigation and remediation.

To drive these improvements, the last section of this paper is separated into two key areas. The first section provides recommendations for software manufacturers to improve their OSS consumption and supply chain security. Aligned with these recommendations, the next section explores potential strategies for future policies and regulations. It is important to note that these recommendations are not a wish list–they are realistic and based on existing best practices utilized in supply chain management across various sectors. Each recommendation represents a reality that can be achieved today through best practices, processes, and tooling.

Software Manufacturers Recommendations

Build security into software products by design.

Customers expect the software products they purchase and use to be secure and safe. The federal government has made it clear that software manufacturers are responsible for ensuring that expectation is met by design. Meeting that expectation means software manufacturers must actively ensure those parts are free from defects. In alignment with the first principle borrowed from Deming's supply chain management best practices, reducing defects requires attention to open source consumption at the beginning of software development. Waiting until after a product is shipped to identify vulnerabilities is too late. Instead, software manufacturers should create an environment that supports their development teams with the information and context needed to make the best choices when they begin writing code.

When tackling this recommendation, it is important to consider existing developer workflows. If approaches are draconian or overly cumbersome, the loss of developer efficiency can discount the reduction in risk and improved OSS consumption processes. In many cases, developers do not look at OSS consumption in the same way as other forms of procurement. Success requires proper strategies to ensure that changes are not arduous and do not add undue friction for development teams.

Finally, consider that not all vulnerabilities are exploitable in every situation, and as such, some products may ship with OSS vulnerabilities that introduce little to no risk. Regardless, it is important to balance expectations against an organization's risk tolerance. Every organization will have a different tolerance, but this is not justification to leave tolerance for risk undefined. Software manufacturers must create policies for OSS consumption that match defined risk tolerance and are integrated throughout the SDLC.

Consume only high-value open source software, components, and projects.

The second principle borrowed from Deming focuses on identifying suppliers that produce the best parts and using them exclusively. While "best" can be highly subjective, software manufacturers should prioritize OSS that consistently provides measurable value to the organization and is updated and supported by an active group of contributors. Measuring value starts with identifying known vulnerabilities and includes criteria like update frequency and how long it takes a project or contributor to fix a vulnerability, among others. Once the best option is identified, manufacturers should use it exclusively. For example, utilizing a single logging framework like Log4j across all software products. In doing so, software manufacturers can reduce their overall risk surface and focus on the OSS that best meets their needs.

According to Deming, the relationship with a supplier is as important as the goods produced. Though OSS is not "supplied" in a traditional sense, the principle around relationships stands. Software manufacturers should contribute back to open source projects as much as possible, especially for remediating vulnerabilities. This should be considered an investment in the long-term availability and quality of the product and can reduce downstream risks associated with future vulnerabilities.

50 "Review of the December 2021 Log4j Event."

51 "National Cybersecurity Strategy," The White House, March 2023, https://www.whitehouse.gov/wp-content/uploads/2023/03/National-Cybersecurity-Strategy-2023.pdf; "Secure Software Development Attestation Form Instructions," CISA, March 2023, https://www.whitehouse.gov/wp-content/uploads/2023/03/National-Cybersecurity-Strategy-2023.pdf.

52 Trey Herr et al., "Buying Down Risk: Cyber Liability," Atlantic Council, May 3, 2022, https://www.atlanticcouncil.org/content-series/buying-down-risk/cyber-liability/.

Continuously track, monitor, and improve the security of open source software being consumed.

The last principle borrowed from Deming directs software manufacturers to continuously track, monitor, and improve their OSS consumption. While this is not manually feasible for software, many modern tools support the creation of an organization-level manifest. This manifest should include all OSS consumed within the context of specific products and across every stage of the SDLC. Having a list of OSS and where it is used is only the first step. Key criteria such as known vulnerabilities, age, mean remediation time, and other metadata must be tracked to improve choices and aid decision-making.

Finally, software manufacturers must work at an organizational level to limit their exposure to vulnerabilities, which starts with establishing an OSS consumption policy aligned with the organization's risk tolerance. This policy provides the foundation for broader, organizational-level governance of OSS consumption. The intent is not to create a list of approved components and reprimand teams when an unapproved component is discovered. Instead, OSS consumption policy should guide decision-making for OSS across the SDLC. More importantly, an organizational policy for open source should be used to educate teams and improve OSS consumption in the long term.

Recommendations for policymakers

Hold software manufacturers responsible and accountable via a national standard of care.

In "Tragedy of the Digital Commons" Sharma argues that "Open source defects should be governed the same way product defects are: when a defect in a product injures a consumer, the law holds every commercial link in the supply chain capable of having identified and remediated the defect accountable."[53] This view represents an expectation of due care by software manufacturers no different than for manufacturers of any physical product. As Sharma points out, software manufacturers have long been able to disclaim liability under outdated interpretations of contract law.[54] Legal tools like end-user license agreements (EULAs) typically contain indemnity clauses protecting software manufacturers from liability. However, the National Cybersecurity Strategy (NCS) hopes to change this with its call to "hold the stewards of our data accountable … reshape laws that govern liability for data losses and harm caused by cybersecurity errors, software vulnerabilities, and other risks created by software and digital technologies."[55]

Software manufacturers' lack of responsibility runs counter to recommendations from Deming's first principle and CISA's guidance that products should be secure by design.[56] While the National Cybersecurity Strategy Implementation Plan's (NCSIP) Strategic Objective 3.3.1 calls for the development of a software security liability framework and mentions a standard of care, it does not offer substantive details.[57] To address this conflict, future policy should solidify the recommendations from the NCS and NCSIP by creating a national standard of care that enumerates the responsibility of software manufacturers to:

1. Identify and evaluate OSS used across their portfolio of products
2. Catalog collected data for OSS
3. Define OSS policies and governance standards
4. Implement continuous vulnerability tracking and monitoring capabilities across the SDLC
5. Quickly and directly disclose and remediate vulnerabilities

These capabilities would improve security across the board for both OSS and proprietary component software.

Require software manufacturers to demonstrate their approach to vetting OSS used in their products.

Many software manufacturers have no standards for their OSS consumption. The White House Office of Management and Budget (OMB) sets vendor requirements for federal agencies looking to acquire software products based on standards like NIST's Secure Software Development Framework (SSDF).[58] To qualify, a vendor must submit a form attesting to the implementation of the required best practices for the software they provide. However, the SSDF has limitations. While the framework provides guidance for software manufacturers to "define security-related criteria for selecting software," it provides no details as to the potential criteria to be used beyond requiring third parties to attest they meet defined standards. Even under this paradigm, neither the standards nor attestation requirements indicate a software manufacturer's approach to OSS consumption beyond technical acquisition (repository, download location, etc.).

According to Deming's second principle, software manufacturers should use the best OSS. The intent of this recommendation is not to define "best." More important is the process software manufacturers use to evaluate the OSS they

53 Sharma, "Tragedy of the Digital Commons."

54 "CISA Director Easterly Remarks at Carnegie Mellon University," February 27, 2023, https://www.cisa.gov/cisa-director-easterly-remarks-carnegie-mellon-university.

55 "National Cybersecurity Strategy."

56 "Shifting the Balance of Cybersecurity Risk: Principles and Approaches for Security-By-Design and -Default."

57 "National Cybersecurity Strategy Implementation Plan", The White House, July 2023, https://www.whitehouse.gov/wp-content/uploads/2023/07/National-Cybersecurity-Strategy-Implementation-Plan-WH.gov_.pdf.

58 "Secure Software Development Framework," NIST, January 10, 2023, https://csrc.nist.gov/Projects/ssdf.

consume and how it is measured against their risk tolerance, both of which should be disclosed to customers. A good foundation for these processes can be found in Open Source Security Foundation's Open Source Consumption Manifesto, which calls upon all software manufacturers to commit to improving their consumption of OSS through fifteen principles and best practices.[59] With these considerations in mind, future policy should require all software manufacturers to follow expanded standards for OSS software consumption, including evaluation criteria, applied decision-making best practices, and detailed process descriptions. Software procurement and acquisition requirements for vendors and contractors at the federal level should be expanded to include qualified details of a software manufacturer's organizational OSS consumption policy, including specifics on the criteria, processes, and tools used when consuming OSS. As outlined in the NCSIP's Strategic Objective 3.5.2, the False Claims Act provides an enforcement mechanism to ensure truthful attestation, holding software manufacturers accountable to these expanded requirements.

Drive software manufacturers to continuously track, monitor, and improve OSS consumption

Strategic Objective 3.3.3 of the National Cybersecurity Strategy Implementation Plan focuses on CVDs.[60] While imperfect, the current CVD process works well when communicating from upstream (OSS) to downstream (software manufacturers), but only in scenarios where a software manufacturer continuously tracks, monitors, and improves their consumption of OSS. In cases where this is not done and a critical vulnerability, like Log4shell, fails to make headlines, many software manufacturers do not know the potential risk they create for their customers. This is the exact scenario we see based on research demonstrating a significant proportion of OSS is downloaded with known vulnerabilities while non-vulnerable versions are available.

The third principle adapted from Deming recommends an approach to continuously track, monitor, and improve OSS consumption. This will result in a more proactive response, better communication with customers, and closer alignment with the intent of the recall process utilized by automotive manufacturers. To meet this recommendation in the short term, acquisition and procurement policy should require manufacturers to demonstrate CVD processes for responding to and mitigating OSS with known, critical vulnerabilities in their software products. In the longer term, the requirement should evolve to demonstrate alignment with more robust vulnerability reporting and disclosure—for example, the National Institute of Standards and Technology (NIST) Vulnerability Disclosure Report, highlighted in NIST's Cybersecurity Supply Chain Risk Management Practices for Technology and Management (C-SCRM) and the Secure Software Development Framework,[61] or utilization of the Vulnerability Exploitability eXchange (VEX), currently led by CISA.[62]

Beyond controls at the federal government level, disclosure and recall processes for software manufacturers should be aligned with a defined standard of care. Combining these approaches provides a more robust mechanism to drive data security and safety standards for software manufacturers in specific industries, such as financial services and the healthcare sector. Recently proposed regulation from the Securities and Exchange Commission (SEC) recommends new cybersecurity risk management and governance standards, including a requirement for public software manufacturers to adopt a more detailed disclosure process.[63] In many ways, the SEC has taken this a step further by defining responsibility for public companies and other businesses within their scope of regulation through their proposed requirement that a demonstration of those processes be provided.[64] Expanding this requirement to adopt disclosure mechanisms specific to OSS vulnerabilities would require software manufacturers to track, monitor, and improve open source consumption in line with the SEC's more general cybersecurity requirements.

BRINGING IT ALL TOGETHER

The road to improvement is paved by lessons learned. Change is hard, and defects are a constant threat to delivering safe and secure products. For manufacturers, minimizing defects is attached to the longevity and reputation of their enterprise, which often hinges on avoiding liability as well. In the past, software manufacturers could avoid liability by delivering products without the same standard of care as their manufacturing peers; that option is ending. Today, the US government, along with governments worldwide, has begun implementing policies and regulations to hold manufacturers responsible for safe and secure software. But gaps remain.

The goal is simple: software manufacturers must build security into software products by design, choose the best suppliers, and track and monitor where those parts are used. This squarely places the responsibility for open source consumption and software supply chain security on manufacturers. To address this more holistically, this paper has focused on the importance of OSS consumption as a critical piece to better software supply chain management. The provided recommendations provided are time-tested approaches deployed in traditional automotive manufacturing. These

59 "The Open Source Consumption Manifesto," OpenSSF EUEG, August 24, 2023, https://github.com/ossf/wg-endusers/tree/main/MANIFESTO.

60 "National Cybersecurity Strategy Implementation Plan."

61 "Shifting the Balance of Cybersecurity Risk: Principles and Approaches for Security-By- Design and -Default;" Murugiah Souppaya, Karen Scarfone, and Donna Dodson, "Secure Software Development Framework (SSDF) Version 1.1," NIST Special Publication 800-218, February 2022, https://doi.org/10.6028/nist.sp.800-218.

62 Tom Alrich, "VEX (Vulnerability Exploitability eXchange): Purpose and Use Cases," FOSSA, June 08, 2023, https://fossa.com/blog/vulnerability-exploitability-exchange-vex-purpose-use-cases/

63 "Fact Sheet: Public Company Cybersecurity; Proposed Rules," SEC, 2022, https://perma.cc/5P34-UV92.

64 Maia Hamin, "Who's Afraid of the SEC?" Atlantic Council DFRLab, June 14, 2023.

ideas are not new–they represent what software manufacturers should be doing already. Every recommendation is built on the same principles.

The aim is not to stifle innovation. Instead, it is to unwind an approach that sidesteps responsibility for due care and to encourage proactive, communicative processes. Policy improvements and expanded guidance provide an opportunity to help software manufacturers improve their responses to defects and communication with customers through recall-like capabilities. What is presented in this paper is a win-win. These principles simultaneously help improve open source and proprietary software supply chains while reducing the overall impact and cost of critical vulnerabilities like Log4shell altogether.

ACKNOWLEDGMENTS

The authors would like to thank Deborah Bryant, Maia Hamin, Stewart Scott, Shane Miller, Jonathan Meadows, Christopher Robinson, and several individuals who will remain anonymous for their feedback on earlier versions of this document, as well as individuals who attended a Chatham House rule workshop on the paper. In addition, the authors would like to thank Anais Gonzalez and Donald Partyka for their layout of this issue brief.

ABOUT THE AUTHORS

Jeff Wayman has spent more than a decade leading digital content and community teams across OSS Security, DevOps, and DevSecOps roles. In his current position, he guides OSS security thought leadership and content strategy for Sonatype. Jeff promotes OSS security awareness through his work with the OpenSSF End Users Working Group and his contributions to the Atlantic Council's Open Source Policy Network. Jeff is pursuing an MBA at the Gies College of Business at the University of Illinois, Urbana-Champaign, focusing on Digital Marketing and Strategic Innovation.

Brian Fox, Sonatype co-founder and CTO, is a Governing Board member for the Open Source Security Foundation (OpenSSF), a member of the Apache Software Foundation, and former Chair of the Apache Maven project. As a direct contributor to the Maven ecosystem, including the maven-dependency-plugin and maven-enforcer-plugin, he has over twenty years of experience driving the vision behind the project, as well as developing and leading the development of software for organizations ranging from startups to large enterprises. Brian is a frequent speaker at national and regional events including Java User Groups and other development-related conferences.

Atlantic Council

Board of Directors

CHAIRMAN
*John F.W. Rogers

EXECUTIVE CHAIRMAN EMERITUS
*James L. Jones

PRESIDENT AND CEO
*Frederick Kempe

EXECUTIVE VICE CHAIRS
*Adrienne Arsht
*Stephen J. Hadley

VICE CHAIRS
*Robert J. Abernethy
*Alexander V. Mirtchev

TREASURER
*George Lund

DIRECTORS
Stephen Achilles
Gina F. Adams
Timothy D. Adams
*Michael Andersson
Barbara Barrett
Colleen Bell
Sarah E. Beshar
Stephen Biegun
Linden P. Blue
Adam Boehler
John Bonsell
Philip M. Breedlove
Richard R. Burt
*Teresa Carlson
*James E. Cartwright
John E. Chapoton
Ahmed Charai
Melanie Chen
Michael Chertoff
*George Chopivsky
Wesley K. Clark
*Helima Croft
*Ankit N. Desai
Dario Deste
Lawrence Di Rita
*Paula J. Dobriansky
Joseph F. Dunford, Jr.
Richard Edelman
Thomas J. Egan, Jr.
Stuart E. Eizenstat
Mark T. Esper
*Michael Fisch
Alan H. Fleischmann
Jendayi E. Frazer
Meg Gentle
Thomas H. Glocer
John B. Goodman
*Sherri W. Goodman
Marcel Grisnigt
Jarosław Grzesiak
Murathan Günal
Michael V. Hayden
Tim Holt
*Karl V. Hopkins
Kay Bailey Hutchison
Ian Ihnatowycz
Mark Isakowitz
Wolfgang F. Ischinger
Deborah Lee James
*Joia M. Johnson
*Safi Kalo
Andre Kelleners
Brian L. Kelly
Henry A. Kissinger
John E. Klein
*C. Jeffrey Knittel
Joseph Konzelmann
Keith J. Krach
Franklin D. Kramer
Laura Lane
Almar Latour
Yann Le Pallec
Jan M. Lodal
Douglas Lute
Jane Holl Lute
William J. Lynn
Mark Machin
Marco Margheri
Michael Margolis
Chris Marlin
William Marron
Gerardo Mato
Erin McGrain
John M. McHugh
*Judith A. Miller
Dariusz Mioduski
*Richard Morningstar
Georgette Mosbacher
Majida Mourad
Virginia A. Mulberger
Mary Claire Murphy
Julia Nesheiwat
Edward J. Newberry
Franco Nuschese
Joseph S. Nye
Ahmet M. Ören
Sally A. Painter
Ana I. Palacio
*Kostas Pantazopoulos
Alan Pellegrini
David H. Petraeus
*Lisa Pollina
Daniel B. Poneman
*Dina H. Powell McCormick
Michael Punke
Ashraf Qazi
Thomas J. Ridge
Gary Rieschel
Charles O. Rossotti
Harry Sachinis
C. Michael Scaparrotti
Ivan A. Schlager
Rajiv Shah
Gregg Sherrill
Jeff Shockey
Ali Jehangir Siddiqui
Kris Singh
Varun Sivaram
Walter Slocombe
Christopher Smith
Clifford M. Sobel
Michael S. Steele
Richard J.A. Steele
Mary Streett
Nader Tavakoli
*Gil Tenzer
*Frances F. Townsend
Clyde C. Tuggle
Melanne Verveer
Charles F. Wald
Michael F. Walsh
Ronald Weiser
*Al Williams
Ben Wilson
Maciej Witucki
Neal S. Wolin
*Jenny Wood
Guang Yang
Mary C. Yates
Dov S. Zakheim

HONORARY DIRECTORS
James A. Baker, III
Robert M. Gates
James N. Mattis
Michael G. Mullen
Leon E. Panetta
William J. Perry
Condoleezza Rice
Horst Teltschik
William H. Webster

**Executive Committee Members*

List as of July 28, 2023