MITRE | SOLVING PROBLEMS FOR A SAFER WORLD®

LOG4SHELL AND ENDEMIC VULNERABILITIES IN OPEN SOURCE LIBRARIES

David Wilburn and Charles Schmidt

 #22-0797 3-10-2022

Executive Summary

The recent disclosure of a series of vulnerabilities in log4j, and their subsequent widespread exploitation, led to many frantic weeks as cybersecurity researchers and defenders sought to stem attacks using this vulnerability. One factor that contributed to the magnitude of the exploit's impact was the degree to which the log4j libraries had been incorporated into numerous software products and projects, which meant that many software products were vulnerable. The number of impacted products, coupled with challenges in applying fixes, mean the log4j vulnerability (known as log4shell) will remain in the global software ecosystem for a long time.

We use the phrase "endemic vulnerability" to describe this situation, where a vulnerability continues to be found and exploited across the global internet within old and new software products long after it has been identified and patches made available. Stakeholders across the software development community, tech industry, and government must act to address and operate in an internet with endemic risks.

Open source software underlies many everyday technology products that we rely on and take for granted, often in ways that are subtle or invisible to users. Open source software libraries are used in many places: by for-profit firms as components of commercially marketed hardware, software, and services, as well as for in-house, custom-developed software used by governments and companies. While this accelerates innovation, any vulnerabilities in these libraries create the conditions for endemic vulnerabilities that pose long-term risks.

To address the challenges of endemic vulnerabilities, MITRE recommends specific actions by key stakeholders across the public and private sectors:

- **The U.S. government** should identify and provide resources to improve critical open source software technology through accessible grant programs that focus on security through collaboration and cooperation with open source software projects.
- **The software industry and companies procuring software-based solutions** should adopt technologies such as Software Bill of Materials to improve transparency of what software libraries their products use and depend upon. This allows developers and users to more quickly identify and respond to vulnerabilities in underlying software components.
- **IT enterprises** should harden their networks with layered defenses and adopt an "assume breach" mentality. These actions should include outbound network filtering, micro-segmentation strategies derived from zero trust architectures, improved monitoring, and exercising of vulnerability and incident response procedures.

While these steps will not eliminate the presence of endemic vulnerabilities in the software ecosystem, they will help reduce such vulnerabilities and help enterprises to operate more safely in their presence.

Endemic Vulnerabilities in the Global Software Ecosystem

Open source software underpins much of the technology we use today. One study of software codebases sampled from a wide range of industries found that 98% contained open source code modules [1]. This software represented industries ranging from retail and e-commerce, to aviation, education, and telecommunications and wireless. There is a good reason for this prevalence: open source libraries are built to respond to key needs their authors have seen, and are designed from the beginning to be easy to integrate into software. Other open source projects, commercial software developers, and custom applications for enterprises and government make frequent use of these libraries. Especially useful libraries can have a large footprint across the global software ecosystem.

In general, use of these libraries is a very good thing – it accelerates development, increases standardization, and allows software developers to leverage the expertise of other specialists that they might not otherwise be able to access. However, when one of these widely used code libraries has an exploitable vulnerability, the security implications can be wide-reaching and long-lived. In particular, such vulnerabilities have a high probability of becoming "endemic." An endemic vulnerability is one that persists in the global software ecosystem long after its identification and the publication of fixes. Vulnerable open source software libraries are particularly susceptible to creating endemic vulnerabilities because it can be harder to identify vulnerable products that incorporate them, and harder to patch products even when they are known to be vulnerable. As a result, the vulnerabilities can still appear in new products months and years after non-vulnerable versions of the relevant library are available. The recent log4shell vulnerability is an excellent example of why such vulnerabilities can have such longevity.

Log4shell (CVE-2021-44228)

The Apache Software Foundation's (ASF) log4j is an open source logging support library for Java applications that has been under development since the 1990s [2]. Its utility is attested to by the significant number of software products and projects that incorporate its libraries, estimated by Google researchers to stretch into the tens of thousands [3].

In July 2013, an update was made to support Java Naming and Directory Interface (JNDI) lookups in log4j at the request of a log4j user [4]. JNDI is a Java feature that allows software applications to retrieve data and software instructions from remote sources that are not part of the installed application. There are legitimate purposes for using this function in controlled contexts, such as loading dynamic configuration. However, if the inputs to these functions are not adequately controlled, attackers can use JNDI to cause a software application to leak sensitive configuration details, or they can retrieve and execute software instructions from a site they control. The potential risks of using JNDI have been noted on multiple occasions [5] [6] and previously contributed to at least one earlier vulnerability and suspected state-sponsored cyberespionage campaign [7]. Nonetheless, JNDI continues to be used in software applications due to the capabilities it provides to developers.

In late 2021, an Alibaba security researcher discovered that log4j's JNDI functionality was vulnerable to exploitation and reported this to the ASF's log4j developer team [8]. The public disclosure and subsequent widespread exploitation

of this vulnerability, designated as CVE-2021-44228 and known within the cybersecurity community as "log4shell" [9], capped off what has been widely described as an annus horribilis [10] of vulnerability and incident response within the cybersecurity community. The log4j developers quickly released updates mitigating the log4shell vulnerability and several related security vulnerabilities [8]. However, efforts by Google, CISA, and other researchers have uncovered potentially thousands of impacted applications across the open source community, commercial software industry, and product vendors of internet of things (IoT), industrial, and operation technologies [3] [11] [12] [13]. A race soon ensued, with software developers, vendors, IT administrators, and network defenders scrambling to apply updates and respond to intrusions on the one side, and a broad range of adversaries, including pranksters, financially motivated criminals, and suspected state-sponsored actors, on the other [14].

Containing the log4shell vulnerability is quite challenging. The patches ASF developed only fix the log4j library. The open source, commercial, and in-house software applications that use log4j require their own security updates to address the vulnerability. Released patches must then be installed by enterprises and users. Unfortunately, the breadth of log4j's deployment across so many applications throughout the internet – many of which may be abandoned by their developers, unmanaged by their users, or simply lack the ability to be patched (as is the case with some IoT and embedded systems [15]) – guarantees that log4shell will remain an endemic vulnerability and pose continuing risk to internet users for years to come.

Because organizations do not install log4j by itself, many will be unaware of its existence in their enterprise, embedded in software products that employ this library. For the same reason, there remains an ongoing risk of software developers incorporating libraries that indirectly employ vulnerable versions of log4j into new software products. Exploitation paths for the log4shell vulnerability are insidious and unpredictable in nature, sometimes employing indirect attack vectors. Any situation where a vulnerable application's logging capability records a message that an attacker can influence has the potential to allow data leakage or execution of an attacker's instructions, and influencing log messages is often quite easy. Additionally, vulnerable backend log processing infrastructure can be compromised even when it is far from public-facing systems, and even when the public-facing systems themselves are fully secure. The ease and severity of the log4shell attack, combined with the widespread use of log4j across numerous software products and the unpredictability of the attack surface, created a situation that arguably deserves to be ranked among the worst software vulnerabilities ever.

Endemic Vulnerabilities and Open Source Libraries

Most software vulnerabilities become endemic to an extent. A 2020 study of the top routinely exploited vulnerabilities for that year found that only a third of the vulnerabilities were discovered in 2020, with one of the top-exploited vulnerabilities dating back to 2017 [16]. Similarly, endemic vulnerabilities are not limited to open source libraries, with many vulnerabilities in proprietary software remaining present in the global software ecosystem for many years. However, vulnerabilities in open source software libraries have been in the

news in the last few years due to the number of otherwise unrelated products that can be impacted. Heartbleed in 2014 (CVE-2014-0160), GHOST in 2015 (CVE-2015-0235), and now log4shell are all examples of vulnerabilities being discovered and exploited in libraries that were incorporated into a wide range of products.

Endemic vulnerabilities are now a fact of life in the global software ecosystem. The presence of known but unpatched vulnerabilities provides plentiful fodder for malicious actors, with those actors often creating collections of compromised machines (often called "botnets") to support distributed denial-of-service or other attacks on various targets. Endemic vulnerabilities can also persist when careless developers use old, vulnerable software libraries in new products. Today's IT enterprises need to operate in a world where endemic vulnerabilities can create threats both inside and outside their networks. While little can be done to eliminate vulnerable software globally, there are steps that can help reduce the presence of new vulnerabilities and help enterprises be more resilient in the face of endemic vulnerabilities.

Addressing the Challenge

To address these challenges, stakeholders across multiple areas of influence must take action. This includes efforts to address the security of open source software libraries at their source, to enable better vulnerability management and response when they are procured and fielded, and to better constrain and respond to adversaries in the event of later compromise. Government, industry, and IT enterprises need to address the issue of endemic vulnerabilities in three different ways: preemptively by reducing the presence of such vulnerabilities, in immediate response to vulnerability disclosure through better tracking of vulnerable libraries, and after a vulnerability has become endemic by better equipping enterprises to operate with resiliency.

U.S. Government Investment in Open Source Software Security

The U.S. government can help secure the open source software projects that are most depended upon commercially and by national interests. The government should commission a study to identify open source software most in need of investment. This should consider the prevalence of the software within the U.S., especially its use within government and critical infrastructure, as well as the software's potential attack surface. Lessons can be taken from past efforts by the Open Source Security Foundation (OpenSSF), as well as recent research by Google and others identifying complex software dependency chains [3] [17].

The U.S. government should also develop a grant program to fund security improvements to open source software libraries, with grant funding prioritized towards open source software projects deemed of greatest importance and in greatest need of security improvements. Such a grant program should be accessible to open source software development and security teams, and apply lessons learned from previous successful efforts such as DARPA's Cyber Fast Track program. Individual or small teams of security-focused developers would be funded to identify weaknesses in critical open source software, working collaboratively with the owners of open source software projects to directly improve the software in question through code contributions and other forms of assistance.

The U.S. government should also encourage and incentivize commercial industry and allied nations to contribute to the open source software projects they depend upon. The Heartbleed vulnerability in OpenSSL triggered tech industry and community action through the Core Infrastructure Initiative, and later the OpenSSF [17]. More recently, Microsoft and Google have committed to supporting open source security through recent

donations to the OpenSSF's Alpha-Omega Project, which aims to identify and fix vulnerabilities within open source software projects [18]. Likewise, the U.S. government can make it easier for innovations and improvements to open source software libraries developed under existing grants to be contributed back to open source projects. Encouraging and incentivizing activities like these adds more resources to the overall effort to better secure these critical software resources.

As a further enabler, the U.S. government should invest in tools and technologies that will help open source developers ensure greater security in their own software and libraries. This includes supporting tools that provide insight into the pedigree of each project, what development best practices are not being followed, known vulnerabilities, and code-level weaknesses that can compromise the security of software projects. Analysis tools exist today that can help open source developers identify these risks; one such tool is free-of-charge Coverity, which DHS helped kick off in 2006 [19].

But vulnerability detection remains challenging – potential vulnerabilities can slip past some scanning tools, while false positives can create an unnecessary burden on developers. Additional R&D is needed to improve coding technology and scanning capabilities that can help eliminate dangerous weaknesses from software before it is used in operational settings. This will require new thinking in the areas of programming languages, supportive secure coding libraries, education and guidance, and analysis techniques that can be trusted to deliver accurate results. Government investment in such technologies will further help to shore up the global software ecosystem.

The main impact of these actions will be greater confidence in the security of widely used open source software libraries. Efforts such as these can help identify and remove vulnerabilities from libraries before they are widely deployed in tools. A contributing factor to log4shell's widespread presence was the vulnerability's existence for more than eight years before it was detected. Had teams recognized and remediated the vulnerability earlier, its footprint would have been far smaller, with fewer vulnerable applications. Minimizing a vulnerability's footprint not only reduces the number of applications defenders need to remediate, but it can make exploiting the vulnerability less attractive to attackers, who will have fewer targets and may see less return (monetary, strategic, or other) for their efforts.

Although these recommended programs should reduce the number of applications impacted by vulnerable libraries, they are unlikely to bring this number to zero. For this reason, further steps are needed to help developers and defenders respond when vulnerable libraries do enter the software ecosystem.

Improved Transparency of Included Software Libraries

A Software Bill of Materials (SBOM) can be used to support the response to the discovery of vulnerable software libraries. SBOMs are data records that identify specific make, model, and version of software products and libraries. Relevant to the case of log4shell and other vulnerable libraries, SBOMs can also be used to record the set of packages and libraries a given software product uses, including packages in multiple levels of dependencies, which can provide valuable information to enterprise operators.

For log4shell, a set of SBOMs for an enterprise's software inventory would help address whether vulnerable libraries have been included deep within the software dependency tree of an application, and help the enterprise better plan its response

to the announcement of a vulnerability. SBOMs would also help software developers avoid endemic vulnerabilities in software packages they might otherwise be tempted to incorporate in their projects. Today, numerous software development tools can generate SBOMs automatically, significantly reducing the hurdles for a developer to create and distribute these records with their products and patches [20] [21]. Recently, implementation of SBOMs became a requirement within the federal government under Executive Order 14028, issued in May 2021 [22]. Given this and the benefit SBOMs provide in tracing vulnerable products and libraries, government and commercial customers should require industry support for SBOMs from their software vendors.

Over the past few years, there have been many calls for the software industry to support SBOMs [20] [22] [23], but SBOM publication is only the first step. SBOMs delivered with a software product need to be updated whenever the product is updated and patched, so they stay in sync with the actual composition of the evolving software product. At the same time, authors of software inventory, vulnerability management, and software development tools need to support consumption of SBOMs by their tools. These steps help ensure that SBOMs are not only present in the software ecosystem, but that they provide real and ongoing value.

SBOMs do not prevent vulnerabilities like log4shell from existing. However, investment in industry adoption of SBOMs will make it easier for enterprises to respond to these vulnerabilities and reduce the chance of the same vulnerability appearing in new products through careless inclusion of vulnerable libraries.

Network Hardening to Mitigate Exploitation Risks

Because endemic vulnerabilities will remain a danger for the foreseeable future, organizations must take aggressive measures to keep their enterprises resilient, adopting an "assume breach" mentality that treats all networked devices and their software with suspicion of potential compromise.

An important element of network hardening is greater network segmentation, isolation, and filtering. Many server-oriented software applications, including those impacted by log4shell, have limited requirements for outbound network connectivity. Organizations should identify outbound network requirements for server applications and enforce greater restrictions with firewalls and other network security devices. This is not a panacea, especially with vulnerabilities like log4shell that can be triggered through protocols like DNS that are harder to restrict. However, this may constrain future attackers' ability to pull down and execute malware or initiate command and control activity from compromised systems.

Elements of zero trust architecture (ZTA) may also prove useful in containing threat activity. While much of the current ZTA emphasis is on control of desktops and laptops, other aspects of ZTA may be valuable when applied to server-oriented applications and appliances. One such ZTA aspect is micro-segmentation, which can impede adversary lateral movement within an enterprise and thus localize any damage from a compromise. Organizations should apply principles of micro-segmentation by isolating server applications and appliances and enforcing minimized internal network connectivity through internal firewalls and network devices. As with SBOMs, migration towards ZTA is a requirement within the federal government under Executive Order 14028 [22].

Web application firewalls (WAFs) also deserve attention, especially for security vulnerabilities that are exposed through web-based protocols, as is commonly the case for log4shell. WAFs can provide additional traffic inspection and filtering. However, given time, attackers can work around almost any filter rule, so WAFs cannot be deployed in a set-and-forget approach. When deployed carelessly, WAFs can interfere with the functionality of the web sites they are meant to protect, or fail to protect them against the most recent threats. To be most effective, WAFs must be frequently but carefully updated to support prevention and detection rules for emerging threat activity. With log4shell, this became a cat-and-mouse game, with malicious parties developing new attacks and obfuscation techniques, followed by WAF vendors releasing new rules to counter them. Nonetheless, a deployed WAF can effectively prevent and detect some future threats. In addition to deploying WAFs to protect public-facing web services, organizations should develop procedures to more rapidly deploy vendor-published prevention and detection rules during times of emergent threat activity, balancing the risk of service degradation against the risk of compromise.

Even after an initial compromise, these techniques can give defenders a significant initial advantage by blocking, or at least slowing, attacker activities in an enterprise. However, given enough time, adversaries will often find ways around defenses. For this reason, containment capabilities need to be matched by efforts to hunt for adversary activity.

Hunting for Post-Exploitation Activity

There is a common saying within cybersecurity: an attacker only needs to be right once, but defenders must be right all the time – which can be particularly dispiriting in the face of aggressive campaigns. However, the reality is that network defenders have many opportunities to detect common adversary tactics, techniques, and procedures (TTPs) beyond initial exploitation of a vulnerability. Organizations should empower their network defenders to take advantage of these opportunities through a more defensible security posture and better detection and response capabilities. A properly defended and instrumented network should present a figurative minefield to attackers, where any step creates an opportunity for defenders to detect and disrupt.

Organizations should leverage MITRE's ATT&CK® framework or similar frameworks to identify opportunities to detect and respond to common adversary TTPs. Gaps can be filled through better network and endpoint instrumentation, including collection of data sources and logging that may be freely available from operating systems and software applications, and through procurement of additional commercial security solutions. Organizations should also invest in detection engineering to identify potentially malicious activity within collected data sources, leveraging hypothesis-driven hunting for specific adversary TTPs. Many organizations may also benefit from judicious use of deception technologies to create further tripwires within their networks. Organizations should also use techniques such as red teaming, purple teaming, adversary emulation, and tabletop exercises to validate their detection and response capabilities, including re-checking for the presence of known vulnerabilities in new applications within their networks.

Implementing these enterprise practices can help security operators turn the tables on adversaries, making them the ones who must be correct "every time" to avoid detection. These practices help create operational resiliency – allowing enterprises to securely support their missions even if a new vulnerability arises. Combined with the previously mentioned practices to constrain an attacker's activities in an enterprise, active hunt efforts reduce the chance that an initial exploitation will turn into a severe security incident.

Conclusions

Endemic vulnerabilities create an enduring problem within the global software ecosystem. When these vulnerabilities exist in widely used software libraries (like log4j), their impact is both broad and challenging to mitigate, even when fixes have been developed for the underlying vulnerability. The U.S. government has identified the need to address these risks as a national priority, while ensuring the nation can continue to enjoy the significant benefits open source software provides [24]. While there are no panaceas, all stakeholders within the global software ecosystem can play a part in helping to reduce the impact of endemic vulnerabilities through a range of actions.

- Investment in improved security in open source software libraries can help reduce the presence of such vulnerabilities within the software ecosystem.
- Sponsorship of security improvements by governments and commercial entities, in collaboration with open source developers, can help reduce the chance of vulnerable libraries creating endemic vulnerabilities in the first place.
- Investment by government and industry organizations responsible for procuring and fielding software in frameworks like SBOMs can provide greater transparency of what libraries are present in their software, simplify identification of vulnerable software, enable better vulnerability response, and reduce the chance of new software reintroducing old vulnerabilities.
- At the IT enterprise level, organizations can harden networks to constrain and isolate risks of compromise and increase detection and response capabilities to manage residual risk, slow adversaries should they gain an initial foothold, and increase the chance that adversary activities will be detected and blocked before they can cause significant damage.

These actions will not eliminate the threat posed by endemic vulnerabilities like log4shell – no one has the power to do that – but they can reduce the frequency of such vulnerabilities and enable more effective response to vulnerabilities that do arise.

References

1. Synopsys, "2021 Open Source Security and Risk Analysis Report," Synopsys Inc., Mountain View, CA, 2021.
2. The Apache Software Foundation Log4J Project, "Apache log4j 1.2 Release History," 9 June 2012. [Online]. Available: https://logging.apache.org/log4j/1.2/changes-report.html.
3. J. Wetter and N. Ringland, "Understanding the Impact of Apache Log4j Vulnerability," 17 December 2021. [Online]. Available: https://security.googleblog.com/2021/12/understanding-impact-of-apache-log4j.html.
4. The Apache Software Foundation Log4J Project, "JNDI Lookup plugin Support," 17 July 2013. [Online]. Available: https://issues.apache.org/jira/browse/LOG4J2-313.
5. A. Muñoz and O. Mirosh, "A Journey from JNDI/LDAP Manipulation To Remote Code Execution Dream Land," in blackhat USA, Las Vegas, 2016.
6. M. Stepankin, "Exploiting JNDI Injections in Java," 3 January 2019. [Online]. Available: https://www.veracode.com/blog/research/exploiting-jndi-injections-java.
7. J. Tang, "New Headaches: How The Pawn Storm Zero-Day Evaded Java's Click-to-Play Protection," 19 October 2015. [Online]. Available: https://web.archive.org/web/20151226072009/http://blog.trendmicro.com/trendlabs-security-intelligence/new-headaches-how-the-pawn-storm-zero-day-evaded-javas-click-to-play-protection/. [Accessed 30 January 2022].
8. The Apache Software Foundation, "Apache Log4j Security Vulnerabilities," 18 December 2021. [Online]. Available: https://logging.apache.org/log4j/2.x/security.html.
9. F. e. a. Whortley, "Log4Shell: RCE 0-day exploit found in log4j 2, a popular Java logging package," 19 December 2021. [Online]. Available: https://www.lunasec.io/docs/blog/log4j-zero-day/.
10. B. Hounshell, "Pro tip: If you call something an "annus horribilis" instead of a "bad year," it makes you seem smarter," 22 December 2021. [Online]. Available: https://twitter.com/blakehounshell/status/1473665187458785283.
11. CISA et al., "CISA Log4j (CVE-2021-44228) Vulnerability Guidance," [Online]. Available: https://github.com/cisagov/log4j-affected-db. [Accessed 29 01 2022].
12. R. Dsouza, "What actions customers can take to protect, detect, and respond to Log4j vulnerabilities in Operational Technology (OT) and Industrial Internet of Things (IIoT) environments," AWS, 20 January 2022. [Online]. Available: https://aws.amazon.com/blogs/iot/what-actions-customers-can-take-to-protect-detect-and-respond-to-log4j-vulnerabilities-in-operational-technology-ot-and-industrial-internet-of-things-iiot-environments/. [Accessed 8 February 2022].
13. D. Greenfield, "Cybersecurity at the Forefront of Industry Concerns in Early 2022," AutomationWorld, 4 January 2022. [Online]. Available: https://www.automationworld.com/cybersecurity/article/21977403/log4j-cybersecurity-concerns-in-industry. [Accessed 8 February 2022].
14. Microsoft Threat Intelligence Center (MSTIC), "Guidance for preventing, detecting, and hunting for exploitation of the Log4j 2 vulnerability," 11 December 2021. [Online]. Available: https://www.microsoft.com/security/blog/2021/12/11/guidance-for-preventing-detecting-and-hunting-for-cve-2021-44228-log4j-2-exploitation/. [Accessed 30 January 2022].

15. J. Valentino-DeVries, "Rarely Patched Software Bugs in Home Routers Cripple Security," The Wall Street Journal, 18 January 2016. [Online]. Available: http://www.wsj.com/articles/rarely-patched-software-bugs-in-home-routers-cripple-security-1453136285?mod=trending_now_6. [Accessed 1 February 2016].
16. Cybersecurity & Infrastructure Security Agency, "Top Routinely Exploited Vulnerabilities - Alert (AA21-209A)," 28 July 2021. [Online]. Available: https://www.cisa.gov/uscert/ncas/alerts/aa21-209a.
17. The Linux Foundation, "Amazon Web Services, Cisco, Dell, Facebook, Fujitsu, Google, IBM, Intel, Microsoft, NetApp, Rackspace, VMware and The Linux Foundation Form New Initiative to Support Critical Open Source Projects," 24 April 2014. [Online]. Available: https://web.archive.org/web/20140425124353/http://www.linuxfoundation.org/news-media/announcements/2014/04/amazon-web-services-cisco-dell-facebook-fujitsu-google-ibm-intel. [Accessed 2 February 2022].
18. Open Source Security Foundation, "OpenSSF Announces The Alpha-Omega Project to Improve Software Supply Chain Security for 10,000 OSS Projects," 1 February 2022. [Online]. Available: https://openssf.org/press-release/2022/02/01/openssf-announces-the-alpha-omega-project-to-improve-software-supply-chain-security-for-10000-oss-projects/.
19. Synopsis, "About Coverity Scan," Synopsis, [Online]. Available: https://scan.coverity.com/about. [Accessed 2 February 2022].
20. J. Reynolds, "What is a Software Bill of Materials (SBOM)?," Sonatype, 27 September 2021. [Online]. Available: https://blog.sonatype.com/what-is-a-software-bill-of-materials. [Accessed 8 February 2022].
21. GitHub, "CycloneDX / cyclonedx-maven-plugin," GitHub, 2 September 2021. [Online]. Available: https://github.com/CycloneDX/cyclonedx-maven-plugin. [Accessed 8 February 2022].
22. The White House, "Executive Order on Improving the Nation's Cybersecurity," 12 May 2021. [Online]. Available: https://www.whitehouse.gov/briefing-room/presidential-actions/2021/05/12/executive-order-on-improving-the-nations-cybersecurity/.
23. J. McKeon, "Healthcare Sector Spearheads SBOM Adoption to Support Cybersecurity," Health IT Security, 1 February 2022. [Online]. Available: https://healthitsecurity.com/news/healthcare-sector-spearheads-sbom-adoption-to-support-cybersecurity. [Accessed 11 February 2022].
24. The White House Briefing Room, "Readout of White House Meeting on Software Security," The White House, 13 January 2022. [Online]. Available: https://www.whitehouse.gov/briefing-room/statements-releases/2022/01/13/readout-of-white-house-meeting-on-software-security/. [Accessed 11 February 2022].

ACKNOWLEDGEMENTS

The authors extend thanks to their MITRE colleagues, including Daryl Baker, Drew Buttner, Brant Goings, Phil Long, Cathy McCollum, Dave Proulx, Emory Tate, and Craig Weiner for their advice, review, and expertise.

About MITRE

MITRE's mission-driven teams are dedicated to solving problems for a safer world. Through our public-private partnerships and federally funded R&D centers, we work across government and in partnership with industry to tackle challenges to the safety, stability, and well-being of our nation.

MITRE | SOLVING PROBLEMS FOR A SAFER WORLD®