March 9, 2022

Chris Inglis, National Cyber Director
Anne Neuberger, Deputy National Security Advisor for Cyber and Emerging Technology
The White House
1600 Pennsylvania Avenue NW
Washington, DC 20500

Mr. Inglis and Ms. Neuberger,

BSA | The Software Alliance[1] supports your efforts to "improve the security of open source software" and collaborate to "rapidly drive improvements."[2] The best opportunities to improve the security of open source software arise from operational improvements, though there are targeted areas for which policy improvements could also be helpful. BSA looks forward to collaborating to identify these opportunities and advance software security.

BSA is the leading advocate for the global enterprise software industry before governments and in the international marketplace. Its members are among the world's most innovative companies, providing the products and services that power governments and businesses. BSA members are also leaders in cybersecurity, having pioneered many of the software security best practices used throughout the industry today, including The BSA Framework for Secure Software.

BSA continues to support, and has a long history of supporting, robust software security. Indeed, BSA recently recognized robust software security as the first priority in Strengthening Trust, Safeguarding Digital Transformation: BSA's Cybersecurity Agenda.

[1] BSA's members include: Adobe, Alteryx, Atlassian, Autodesk, Bentley Systems, Box, Cisco, CNC/Mastercam, DocuSign, Dropbox, IBM, Informatica, Intel, MathWorks, Microsoft, Okta, Oracle, PTC, Salesforce, SAP, ServiceNow, Shopify Inc., Siemens Industry Software Inc., Splunk, Trend Micro, Trimble Solutions Corporation, Twilio, Unity Technologies, Inc., Workday, Zendesk, and Zoom Video Communications, Inc.
[2] Read out of White House Meeting on Software Security (January 13, 2022), available at https://www.whitehouse.gov/briefing-room/statements-releases/2022/01/13/readout-of-white-house-meeting-on-software-security/.

The "open source community" is large and diverse. It ranges from multinational, enterprise software companies to individual volunteers. Community members include both those who develop and maintain open source software and those who consume open source software. In fact, many community members develop, maintain, *and* consume open source software. Most work in the open source community is done through open source software projects, self-organizing groups of organizations and individuals with self-governance structures used to develop software. Open source software projects may be funded, organized, governed, and sustained by organizations (e.g. governments, multinational corporations, academic institutions, non-profits) or individuals (who themselves may be affiliated or supported by organizations). These projects typically include "leaders" of the project, "committers and contributors" who develop code, and "consumers" who use the code.

As you noted in the readout of your January 13 meeting, "Most major software packages include open source software – including software used by the national security community." Open source software brings unique value, as it facilitates innovation that benefits governments, businesses, and society. Given its breadth of use and the number of volunteers responsible for its development, maintenance, and use, open source software also requires approaches to security that strike the right balance of maintaining the benefits of open source innovation while enhancing the security of products that use open source software. This includes efforts that focus on ways to better integrate security into the development and maintenance of open source software while also enhancing the security of products and services that use open source software. Today, it is not realistic to expect any software to be entirely free of vulnerabilities, especially as malicious actors develop new tactics, techniques, and procedures. Rather, developers of open source software, which includes the US Government, should adopt development processes, product security capabilities, and life cycle management approaches that minimize vulnerabilities and their potential impact, as well as support proactive cybersecurity risk management and consumers of open source software should similarly remain responsible for actively managing software security risks when using open source software.

For years, enterprise software companies and open source software projects have invested heavily in open source software security. Importantly, enterprise software companies typically require their employees tasked with developing open source software to use the same or similar secure software development practices these companies require for their own proprietary software.

The log4j vulnerability demonstrates the unique security challenges of open source software because it implicates a piece of open source software incorporated in thousands of products and services. The software is used to perform a logging function but contains a vulnerability that allowed a malicious actor to perform a remote code execution attack. As this vulnerability, and others before it, demonstrated, patch management can be truly challenging and, as the National Institute of Standards and Technology (NIST) recognizes, "organizations that can minimize the time they spend dealing with patching can use those resources for addressing other security concerns."[3]

[3] NIST notes in SP 800-40, "Timing, prioritization, and testing are intertwined issues for enterprise patch management. Ideally, an organization would deploy every new patch immediately to minimize the time that systems are vulnerable to the associated software flaws. However, in reality, this is simply not possible because organizations have limited resources, which makes it necessary to prioritize which patches should be installed before other patches. Further complicating this is the significant risk of installing patches without first testing them, which could cause serious operational disruptions, potentially even more damaging than the corresponding security impact of not pushing the patches out. Unfortunately, testing patches consumes even more of an organization's limited resources

As with every vulnerability, we need to reflect on what we can learn from it and take proactive steps to apply that knowledge to prevent similar vulnerabilities in the future. One lesson is the importance of continuous risk management including deployment of patches for both cloud-based and on premises software. Such patching practices are relatively common for secure cloud, hybrid-cloud, and on premises software. Whether software is cloud-based or on premises, continuous maintenance is foundational for effective software security risk management.

A second lesson is that patches can generally be applied more effectively and efficiently in the cloud. Consequently, accelerating movement to secure cloud services is likely to further improve the cybersecurity posture of any organization that uses them. This aligns directly with the Executive Order on Improving the Nation's Cybersecurity's direction to "accelerate movement to secure cloud services."

A third lesson is the benefit of using software that is currently maintained (including regular patching). Software – including open source software – that is developed and maintained by contributors that use secure development best practices will produce software with fewer vulnerabilities. Simply put, using unmaintained or out-of-date software may be cheaper in the short run, but will be more costly over time and impede remediation when a vulnerability is discovered. Unmaintained and out-of-date software has unintended negative impacts on the entire cybersecurity ecosystem.

A fourth lesson is that enterprise software companies and open source software projects can usually develop patches quickly, but ensuring consumers apply these patches in a timely manner, particularly for on-premises software, remains a challenge.

Below, we provide 12 aggressive but achievable recommendations, in each of the three areas you identified, for making significant improvements in open source software security.

---

and makes patch prioritization even more important. For patch management, timing, prioritization, and testing are often in conflict. Product vendors have responded to this conflict by improving the quality of their patches and bundling patches for their products. Instead of releasing dozens of patches one at a time over a period of three months, necessitating testing and patch deployment every few days, a vendor might release their patches in a single bundle once a quarter. This allows an organization to perform testing once and roll out patches once, which is far more efficient than testing and rolling out all the patches separately. It also reduces the need to prioritize patches—the organization just needs to prioritize the bundle instead of separately prioritizing each patch it contains. Vendors who bundle patches tend to release them monthly or quarterly, except for cases when an unpatched vulnerability is actively being exploited, in which case they usually issue the appropriate patch immediately instead of delaying it for the next bundle."

Minimizing Vulnerabilities in Open Source Software

1. Developers of open source software (which includes the US Government) should use best practices for developing and assessing software security. Tools such as NIST's Secure Software Development Framework[4] (which maps to the BSA Framework for Secure Software) can be used to help (a) software developers describe the current state and target state of software security in individual software products and services; (b) software developers identify opportunities for improvement in development and lifecycle management processes; (c) software developers, vendors, and customers communicate internally and externally about software security; and (d) software consumers evaluate and compare the security of individual software products and services.

2. Developers and consumers of open source software should invest in the development and maintenance of open source software they use. There are many ways they can contribute including providing direct financial support or other resources or participating in the open source project's community. They should do so, in part, by following or adapting the same best practices they use for proprietary software and practices they prefer or require of their vendors, if applicable.

3. The US Government should require all colleges and universities that receive federal funds and that provide instruction on software development, to include in their software development curriculum appropriate instruction on secure software development processes, secure capabilities, and secure lifecycle management.[5]

4. Developers of open source software that have employees should require their employees responsible for developing software, including open source software, to obtain appropriate training on secure development processes, secure capabilities, and secure lifecycle management. Open source projects should make such training available to their contributors. The specifics of appropriate training should be built on existing documents like NIST's Workforce Framework for Cybersecurity.

5. Developers and consumers of open source software should participate in public-private partnership projects, like the ones NIST runs, that are aimed at implementing and demonstrating secure software development practices. This work should include the development of automated tools by organizations such as the Open Source

[4] Problematically, NIST's recent guidance on software security specifically omits open source software and software developed by the US Government. By not requiring its own software developers to meet the same security requirements it puts on its vendors, the US Government is missing an opportunity to lead by example and instead communicating it is not serious about participating in improving software security. Executive Order (EO) 14038 Section 4e, February 4, 2002, available at https://www.nist.gov/system/files/documents/2022/02/04/software-supply-chain-security-guidance-under-EO-14028-section-4e.pdf.

[5] As the National Security Telecommunications Advisory Committee's (NSTAC) report on Software Assurance in the Information and Communications Technology and Services Supply Chain states, "security is foundational for computer science, so that before something is envisioned, designed, or coded, the developer understands the threats the resulting software needs to meet, architects it to mitigate those threats, and keeps the code free (or largely free) of coding errors that can compromise security." NSTAC Report to the President on Software Assurance (November 2, 2021), available at https://www.cisa.gov/sites/default/files/publications/NSTAC%20Report%20to%20the%20President%20on%20Software%20Assurance.pdf.

Security Foundation, that can be used by the open source community.

Improving the Process for Identifying Vulnerabilities and Developing Patches

6. Developers and consumers of open source software should use best practices for identifying vulnerabilities, coordinating disclosure, and developing patches. Tools such as the NIST Secure Software Development Framework (which maps to the BSA Framework for Secure Software) can be used to help, among other things, communicate realistic expectations regarding the lifecycle of software and to identify software that is unmaintained, out of date, or has known vulnerabilities. Consumers of open source software should actively manage the risks associated with the use of software that is at or nearing end of life.

7. Developers and consumers of open source software should commit to working together to identify and prioritize the security of the most critical open source software components and the most critical open source software platforms, as well as improve automated tools for analysis and testing. This work should ensure that the investment in security will have the greatest possible return.

8. Developers and consumers of open source software should proactively maintain their products and services and have vulnerability identification and management processes that may include periodic automated scans of their software for vulnerabilities contained in up-to-date lists of the most critical software vulnerabilities. While these lists will not identify all vulnerabilities, periodic scanning still presents an opportunity to ensure software does not contain well-known and easily exploitable vulnerabilities.

9. The US Government, working through the General Services Administration (GSA), should ensure that GSA's code.gov builds off and is complimentary to the other actions suggested here, including by sharing GSA work with the broader open source community.

Expediting the Distribution and Implementation of Patches

10. Developers and consumers of open source software should use best practices for distributing and implementing patches. Tools such as the BSA Framework for Secure Software provide value by, amongst other things, helping developers (a) make risk-informed decisions about the prioritization of patches; (b) test patches for functionality and security; (c) notify users of significant security issues when a remediation is in place; and (d) disseminate patches securely.

11. Developers and consumers of open source software should respond to a vulnerability commensurate with the risk it creates.[6]

12. Developers of open source software should have a process for considering whether to push out an available patch outside their normal patching schedules.

[6] The Common Vulnerability Scoring System (CVSS) score is a helpful approximation for a vulnerability's risk but importantly might not always accurately describe the risk a vulnerability creates given its specific deployment and any compensating security controls deployed. Indeed, FedRAMP allows for a cloud service provider to downgrade a vulnerability based on more specific facts.

> Patching outside normal patching schedules is particularly important for patches that address a vulnerability that poses a critical risk for on premises software, for which users are responsible for implementing patches. This process, which need not be public, should consider, amongst other things, whether the Common Vulnerability Scoring System (CVSS) score accurately reflects the risk the vulnerability poses given its specific deployment and any compensating security controls deployed.

The recommendations BSA makes call on both private and public sector to improve the security of their own software and the cybersecurity ecosystem more broadly. When security vulnerabilities arise, are identified, and are exploitable, it is an issue that is not confined to national borders. Governments have a common interest in reducing vulnerabilities that are exploitable by bad actors. We encourage the US Government to work with governments around the world to support and use best practices and encourage the actions noted above.

We look forward to continuing to work together to achieve our shared goals.

Sincerely,

[illegible]

Henry Young
Director, Policy