Atlantic Council

Testimony of

Dr. Trey Herr
Director, Cyber Statecraft Initiative
Atlantic Council

Before the
United States House of Representatives
Committee on Science, Space, and Technology
Subcommittee on Investigations and Oversight & Subcommittee on Research and Technology

"SolarWinds and Beyond: Improving the Cybersecurity of Software Supply Chains"

May 25, 2021

Chairman Foster, Chairwoman Stevens, Ranking Members Obernolte and Waltz, and members and staff of the sub-committees – thank you for the invitation to speak today. We are here in no small part because of the revelations about the Sunburst/SolarWinds campaign. In this instance, the length of time the adversary remained in US networks, public and private, is staggering and suggests incredible amounts of information was likely stolen. The operation of the campaign – as much as is known to the public – appears to have required substantial lead time for reconnaissance against more than one hundred victim organizations. The scale of this event, and its impact on the cybersecurity policies of a new administration, have received widespread appreciation and it is duly warranted.

But even with this large and lengthy an operation, no reports have yet surfaced that the adversary exploited a hitherto unknown vulnerability or unprecedented means of attack. Against SolarWinds, the adversary undermined trust in the software supply chain in a manner observed repeatedly over the past decade. The trend line of these attacks is one that merits attention and no small move toward action.

Software Supply Chain Attacks

Since Ada Lovelace deployed the first computer program on an early mechanical device in the 1840s, software has spread to every corner of human experience. [1] Our watches now have Internet connections, combat aircraft come with more code than computer operating systems, and every organization from the Internal Revenue Service to an Etsy storefront relies on software to serve their customers. No longer confined merely to computers, embedded software now controls the operation of complex power generators, medical hardware, the behavior of automotive brake pedals, and planetary scale datasets. As one commentator put it, "software is eating the world."[2]

With software come security flaws and a long chain of updates from vendors and maintainers. This ongoing maintenance leaves software supply chains messy and in continuous flux, resulting in significant and underappreciated aggregated risk for organizations across the world. Unlike a physical system that is little modified once it has left the factory, software is subject to continual revision through updates and patches.

A software supply chain attack occurs when an attacker accesses and modifies software in the software development supply chain to compromise a target farther down on the chain by inserting their own malicious code. Modern software products contain a vast number of dependencies on other code, so tracking down which vulnerabilities compromise which products is a nontrivial organizational and technical feat. Software supply chain attacks take advantage of established channels of trust, between the user and a vendor or developer, to compromise their targets.

In the Sunburst case, intruders were able to access SolarWinds' build infrastructure, rather than

[1] This section and several portions of the following testimony are drawn from "Breaking Trust: Shades of Crisis across an Insecure Software Supply Chain", Trey Herr, June Lee, Will Loomis, and Stewart Scott - https://www.atlanticcouncil.org/in-depth-research-reports/report/breaking-trust-shades-of-crisis-across-an-insecure-software-supply-chain/

[2] Marc Andreessen, "Why Software Is Eating the World," *Wall Street Journal*, August 20, 2011, https://www.wsj.com/articles/SB10001424053111903480904576512250915629460.

just tacking malware onto a pending update. This difference between update and build is rather like choosing where to attach a bomb to a motorcycle. In this case, instead of adding their malware alongside the software just before being sent to customers, like attaching a sidecar with a bomb inside to a motorcycle, the intruders went further and compromised the company's build infrastructure and source code. The result was like secreting a bomb into the cylinders of the motorcycle's engine before it sold—far more deeply embedded in the resulting device, and thus harder to detect or remove.

In this – SolarWinds was only the most recent in a long line of software supply chain attacks. In the last 10 years, there have been more than 140 attacks, or disclosure of vulnerabilities which could be used in such attacks, on the software supply chain.[3] Of these, *at least* 36 were attacks on software updates, including 15 targeting source code or developer's computers of which nearly half of which were attributed to state actors including many targeting administrative or security tools like the SolarWinds Orion software. These attacks on software updates are important and they emerge as a clear, and unsettlingly consistent trend, in software supply chain attacks from the last decade.

There are several other notable trends including that state actors are behind a significant number of these attacks and both mobile application and open-source software have been successfully targeted as well, at times to great effect.

States have used software supply chain attacks to deliver highly impactful software supply chain attacks, thanks in part to recurring failures by vendors to secure the code-signing process for their products. And while concerns about the real-world ramifications of attacks on firmware, IoT devices, and industrial systems are warranted, these are far from novel threats. Stuxnet and other incidents have had physical impacts as early as 2012. Several of these incidents, like NotPetya and the Equifax data breach in 2017, impacted millions of users, showcasing the immense potential scale of software supply chain attacks and their strategic utility for states.

Since 2010, there have been *at least* 30 different state backed software supply chain attacks from states including Russia, China, North Korea, and Iran as well as India, Egypt, and Vietnam.[4] Within a few months of the public discovery of the Sunburst/SolarWinds campaign, there were reports of three other state backed software supply chain campaigns targeting foreign governments for espionage, with victims located in South Korea, Vietnam, and Mongolia.[5] Each of these targeted deeply privileged programs or widely used and mandated programs—usually at the seams between organizations.

Mobile applications remain a frequent vector for software supply chain attacks, with a quarter all publicly reported incidents impacting app stores since 2010.[6] Mobile application hubs and stores are a popular means of disseminating software supply chain attacks. The stores are a common

[3] The dataset associated with this figure and following breakdown of attack types is available online for perusal or download - https://www.atlanticcouncil.org/resources/breaking-trust-the-dataset/
[4] Ibid; Herr et. al "Breaking Trust"
[5] This section and several portions of the following discussion are drawn from "Broken Trust: Lessons from Sunburst", Trey Herr, Will Loomis, Emma Schroeder, Stewart Scott, Emma Schroeder, and Tianjiu Zhou - https://www.atlanticcouncil.org/in-depth-research-reports/report/broken-trust-lessons-from-sunburst/
[6] Herr et. al "Breaking Trust – The Dataset"

feature of the software ecosystem and are how many users interact with the software supply chain on a regular basis. Attackers can build their own apps, designed to appear legitimate, perhaps providing wallpapers, tutorial videos, or games. For instance, in 2017, the app Lovely Wallpaper hid malware under the guise of providing phone background images. The malware would gain device permissions and charge users' accounts for "premium" services they had not signed up for. Together with fifty other apps hiding the same payload, this attack infected as many as 4.2 million devices, and successors continued to infiltrate the associated app store long after the original offenders were removed. [7]

There is also publicly available evidence that attackers compromise the software used to build mobile software, allowing them to inject malware into legitimate applications as they are created. Compromising development tools used to build apps for those stores provides tremendous scale in a software supply chain attack. One example is the XcodeGhost malware, first detected early in the fall of 2015.[8] Xcode is a development environment used exclusively to create iOS and OS X apps. A version of Xcode found on Baidu Yunpan, a Chinese file-sharing service, came embedded with malicious logic to insert a backdoor in hundreds of applications impacting hundreds of millions of users.[9]

Open-source code was not at the heart of the Sunburst crisis, but it is a critically underdefended part of the software supply chain. Open-source software constitutes core infrastructure for major technology systems and critical software. Attacks and disclosures against open-source libraries have been increasingly frequent in recent years, though whether this is due to improved visibility and reporting, or attacker preferences, deserves further study. In February 2020, two accounts uploaded more than 700 packages to the official RubyGems repository and used typosquatting, naming malware in a format very similar to a legitimate software package, to achieve more than 100,000 downloads of their malware, which redirected Bitcoin payments to attacker-controlled wallets.[10] Many of these attacks remain viable against users for weeks or months after software is patches because of the frequency with which open-source projects patch and fail to notify users. Repositories and hubs can do more to help, providing easy to use tools for developers to notify users of changes and updates and shorten the time between when a vulnerability is fixed, and users notified.

Software supply chain insecurity remains a scourge on industry and the public sector despite billions of dollars in security investment over the last decade. Protecting these supply chains demands more persistent focus on the management of risks in software deployment, not just

---

[7] Check Point, "ExpensiveWall: A Dangerous 'Packed' Malware on Google Play That Will Hit Your Wallet," *Check Point Blog*, September 14, 2017, https://blog.checkpoint.com/2017/09/14/expensivewall-dangerous-packed-malware-google-play-will-hit-wallet/.

[8] Joseph Cox, "Hack Brief: Malware Sneaks into the Chinese iOS App Store," *WIRED*, September 18, 2015, https://www.wired.com/2015/09/hack-brief-malware-sneaks-chinese-ios-app-store/.

[9] FireEye Mobile Team, "Protecting Our Customers from XcodeGhost", *FireEye Blogs*, September 22, 2015, https://www.fireeye.com/blog/executive-perspective/2015/09/protecting_our_custo.html; Claud Xiao, "Malware XcodeGhost Infects 39 iOS Apps, Including WeChat, Affecting Hundreds of Millions of Users", *Unit 42*, September 18, 2015, https://unit42.paloaltonetworks.com/malware-xcodeghost-infects-39-ios-apps-including-wechat-affecting-hundreds-of-millions-of-users/

[10] Catalin Cimpanu, "Clipboard hijacking malware found in 725 Ruby libraries", *ZDnet,* April 17, 2020, https://www.zdnet.com/article/clipboard-hijacking-malware-found-in-725-ruby-libraries/

development. NIST can play a more important role in improving the focus, and efficacy, of this risk management through the provision of technical tools and assistance to encourage the implementation of NIST standards and guidance and by building on existing programs of work in the public and private sectors.

Better Securing the Software Supply Chain

The secure development of software is important but addressing the pace and scale of software supply chain attacks demands we pay more, if not equal attention, to how that software is *deployed* and supported. Dozens of the software supply-chain attacks discovered in the last 10 years target weakly secured code signing certificates, update servers, and other tools for software deployment. NIST can first help by assembling all security controls that impact software deployment from its universe of guidance documents in one place, leading a multi-stakeholder process to work with industry in developing a software supply chain Lifecycle Security Overlay to NIST SP 800-53.[11] The Overlay offers an existing process to collect security controls relevant to a specific topic which would be faster than a new standalone special publication and would directly support the directive to create preliminary guidelines to enhance software supply chain security required by EO 14028 section (4)(c). This effort should wrap in controls the new supply-chain family in 800-53 rev. 5 and best practices collected in the Secure Software Development Framework (SSDF) which includes industry proposals and frameworks from SAFECode, OWASP, and others.[12] This work would build on NIST's expertise and strong network and follows on previous recommendations to anchor technical security obligations in standard-setting organizations. It could also capitalize on industry and non-profit led projects like the Linux Foundation's SigStore – an effort to provide free and more robust digital infrastructure to sign code and audit those signatures.[13]

In addition to any standard documentation and report formats, this overlay, and the associated preliminary guidance from EO 14028, should also be delivered as automated software tools or appropriate source material and references for the vendors of widely used developer tools to integrate these controls and an appropriate auditing framework into their products. At present, far too many cybersecurity regulations and risk management schemes are implemented in PDF and spreadsheets rather than the tools used to build and deploy software. This creates meaningful barriers for developers to implement these controls and users to audit them. Very little is to be gained from another standards document developers have to download in pdf form and make their own determination about how to implement.

NIST will need appropriate resources and support to develop software tools, and appropriately engage with industry, to implement these standards and guidance. This is a question of budget and billets as much as investment in time; NIST can become a more effective software development enterprise including following secure deployment practices like signing and maintaining code available on their website. This automation is particularly important for more rapid and "agile" development projects where software may go through multiple versions in a single day, each requiring these controls to be implemented and checked. GitHub's use of an automated tool called Dependabot to detect and flag vulnerabilities in open-source projects as

[11] Herr, et al., *Breaking Trust*.
[12] NIST, *Secure Software Development Framework*, https://csrc.nist.gov/Projects/ssdf
[13] Linux Foundation, *SigStore*, https://sigstore.dev/what_is_sigstore/

developers integrate them into their code is a good example of taking a best practice and practically implementing it.[14] Automation is the only feasible path to ensure security becomes a baked in component to such a software development and deployment pipeline.

One of the most striking lessons from the Sunburst/SolarWinds campaign, echoed in EO 14028, is the importance of securing cloud computing to the discussion of software supply chains. In developing the supply chain security guidance and preliminary guidelines required by EO 14028, NIST can offer mapping of software deployment controls to popular Software as a Service (SaaS), Platform as a Service (PaaS), and Infrastructure as a Service (IaaS) offerings from commercial vendors.[15] NIST may well require additional resources for a greater volume of work on cloud security and cloud architectures as it could substantially expand upon existing efforts.

This focus on the cloud would support implementation in the private and public sectors. The federal government has an opportunity to enforce software security policies on agencies and departments and audit implementation of these policies in real time directly through the cloud services increasingly found in the .gov and .mil. Many of these policy-enforcement mechanisms and data-collection tools are already "baked in" to cloud services; the challenge is mostly in determining how to take advantage of them 'natively' with NIST standards and guidance. Addressing secure software deployment in cloud environments would help ensure this guidance is relevant for most IT environments outside the public sector, increasing the utility of this line of effort, and would provide deeper technical support for DHS efforts to develop a secure cloud governance framework and migration reference architectures per EO 14028 (3)(c)(ii) and (iii).

The move for NIST to be more involved in developing tools to implement their own standards and guidance, including in cloud services, raises an important question of shared responsibility between NIST and operational security partners, especially DHS' Cyber and Infrastructure Security Agency (CISA). NIST is best positioned to develop these control and map how they could be implemented in different kinds of software. NIST, in partnership with CISA, could also develop use-cases of popular combinations of software where these controls might interact or overlap. But it is CISA, and other cognizant operational security agencies, who should be working to develop specific templates and configuration guides for their customer agencies. NIST is not well positioned to become expert on the unique operating conditions and constraints of every agency. In sum, NIST's role should be expert on the controls, deeply familiar with these software products, and positioned to make recommendations on how they interact with major cloud services and cloud deployment models. CISA's role is to take that guidance and tell agencies how to set their dials and knobs – making specific recommendations on configuration and enforcing broader cybersecurity policies.

---

[14] Joe Uchill, "Microsoft's GitHub adds dependency review to new code submitted from programmers", *SC Magazine*, December 9, 2020, https://www.scmagazine.com/home/security-news/microsofts-github-adds-dependency-review-to-new-code-submitted-from-programmers/; Tammy Xu, "How to Keep Software Dependencies From Becoming Your Downfall", *BuiltIn*, February 18, 2021, https://builtin.com/software-engineering-perspectives/dependabot

[15] For more on what cloud computing is and how cloud services work, see Simon Handler, Lily Liu, and Trey Herr, "Dude, Where's My Cloud? A Guide for Wonks and Users", https://www.atlanticcouncil.org/in-depth-research-reports/report/dude-wheres-my-cloud-a-guide-for-wonks-and-users/

Lastly, while the aspirations of these efforts toward more secure software supply chains are necessarily focused on achieving the best possible outcomes, we must recognize that many organizations who seek to implement this guidance may not have the resources to do so effectively. Wendy Nather, of Duo Cisco, articulated the concept of the cyber poverty line to describe the threshold between organizations of equal intent and motivation to secure themselves but diverging resources and maturity. However, many high performing controls and extensive implementation guides emerge from NIST focused on software supply chain security over the next several years – there will be a population of users and some developers who are unable to effectively implement them.

There are two things NIST can do to address this. First, embrace an emphasis on automation. As much as automating controls and guidance to integrate with standard developer tools will enhance adoption of these best practices, it can also help lower the burden of implementation. Removing the need to translate from a pdf into homemade rules for an integrated development environment (IDE) or organization policy saves time, confusion, and potential mistakes. Second, NIST can work to model the environments and constraints of moderate to low resourced IT security organizations and recommend adaptations of existing guidance to fit. The situation is similar to that found in operational technology and industrial IT environments; resources like network bandwidth and computing power are constrained necessitating changes in how users collect and process data from these systems or apply patches. Such an effort to address the cyber poverty line for supply chain security guidance (and it could be applied to all existing supply chain risk management documentation) would help widen implementation of this work across the private sector and enhance the impact of NIST's expertise and efforts.

Conclusion

Trust in software supply chain security is not built, or broken, in isolation. It would be a mistake to equate software supply chain attacks to a new weapon system in an opponent's arsenal. They are manifestation of opportunity, attacking targets by compromising weaknesses in connected neighbors, vendors, and software dependencies. For the technology industry, the insecurity of the software supply chain is a crisis in waiting. For the national security establishment, it is a crisis realized.

There are opportunities for meaningful progress and NIST can play an important role to better protect the code we have embedded in our daily lives with appropriate investment and a greater focus on automatable guidance, cloud security, and software deployment. Change on this front will demand persistence, at least as much as that of the adversary, if not a measure more.

Thank you again for the opportunity to speak with you today. I look forward to your questions.

###

Dr. Trey Herr is the director of the Cyber Statecraft Initiative under the Scowcroft Center for Strategy and Security at the Atlantic Council. His team works on cybersecurity and geopolitics including cloud computing, cyber effects on the battlefield, the security of the internet, supply chain policy, and growing a more capable cybersecurity policy workforce. Previously, he was a senior security strategist with Microsoft handling cloud computing and supply chain security policy as well as a fellow with the Belfer Cybersecurity Project at Harvard Kennedy School and a non-resident fellow with the Hoover Institution at Stanford University. He holds a PhD in Political Science and BS in Musical Theatre and Political Science.