Atlantic Council
SCOWCROFT CENTER
FOR STRATEGY AND SECURITY

CYBER STATECRAFT
INITIATIVE

BREAKING TRUST: Shades of Crisis Across an Insecure Software Supply Chain

Trey Herr, June Lee, William Loomis, and Stewart Scott

Scowcroft Center for Strategy and Security

*The **Scowcroft Center for Strategy and Security** works to develop sustainable, nonpartisan strategies to address the most important security challenges facing the United States and the world. The Center honors General Brent Scowcroft's legacy of service and embodies his ethos of nonpartisan commitment to the cause of security, support for US leadership in cooperation with allies and partners, and dedication to the mentorship of the next generation of leaders.*

Cyber Statecraft Initiative

*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.*

Atlantic Council
SCOWCROFT CENTER
FOR STRATEGY AND SECURITY

CYBER STATECRAFT
INITIATIVE

BREAKING TRUST:
Shades of Crisis Across an Insecure Software Supply Chain

Trey Herr, June Lee, William Loomis, and Stewart Scott

ISBN-13: 978-1-61977-112-3

Cover illustration: Getty Images/DavidGoh

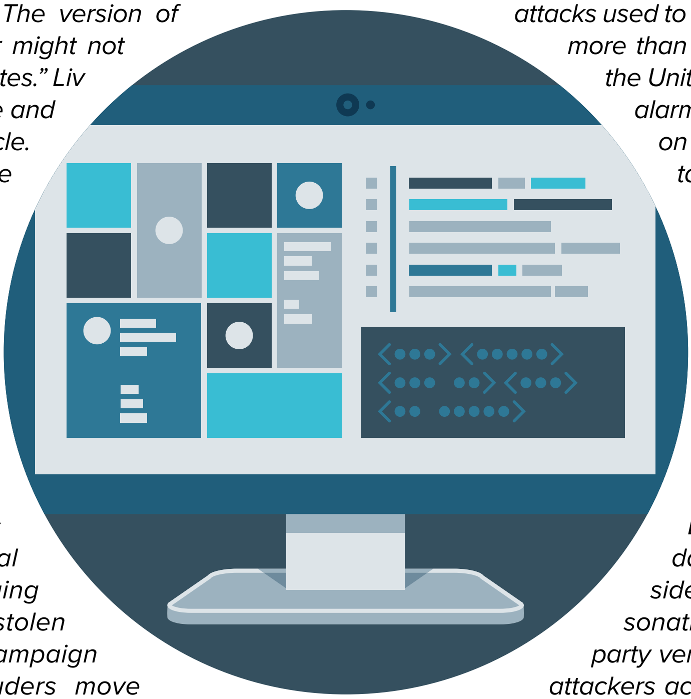
This report is written and published in accordance with the Atlantic Council Policy on Intellectual Independence. The author is solely responsible for its analysis and recommendations. The Atlantic Council and its donors do not determine, nor do they necessarily endorse or advocate for, any of this report's conclusions.

July 2020

Table of Contents

For more on this project and the dataset behind it, please visit us on the web here.

A Supply Chain Story



After a particularly exhausting day at work in February 2017, Liv wraps up her project and prepares to head home. Managing the power grid for a third of the country is high-stakes work and tiring at the best of times. Packing up her bag, she goes to turn off her computer monitor and notices an update waiting patiently on her screen: "Flash Player might be out-of-date. The version of this plug-in on your computer might not include the latest security updates." Liv clicks 'Yes' to begin the update and hurriedly steps out of her cubicle. As she moves quietly down the fall, her laptop fan whirs as it visits specific URLs before downloading a file called "install_flash_player.exe," and, covertly, the Trojan. Karagany.B backdoor.

Liv has no reason to suspect that this software update is different from any other but it allows attackers to quickly install additional tools on her device. Leveraging passwords and usernames stolen through an earlier phishing campaign against Liv's firm, the intruders move quickly across the entire company's network and proceed to take screenshots of sensitive windows and capture images of the company's grid operation control panels. What might have seemed like a harmless software update is actually part of a multiphase campaign that could have allowed attackers to stop the flow of electricity to thousands of businesses and homes in the United States.

This malware isn't fictional. From 2015 to 2017, an extensive campaign called Dragonfly 2.0 saw "Trojanized" software updates alongside phishing emails and watering hole attacks used to gain access to the networks of more than twenty energy sector firms in the United States and in Europe.[1] In an alarming echo of the 2015 attacks on Ukraine's energy grid, the attackers obtained operational control of several firms' networks, giving them the capability to sabotage the energy access of thousands of US users.[2] Using compromised third-party software, attackers gained a foothold in operating systems over the course of the campaign.[3]

Liv wasn't being careless. Updating software regularly is considered best practice. Yet impersonating updates by trusted third-party vendors provided the DragonFly attackers access to major firms in the energy sector. The software supply chain presents a significant source of risk for organizations from critical infrastructure companies to government security agencies but the state of security in this supply chain doesn't match up to the risk. There are opportunities for the policy community and industry to work together to address the problem.

[1] Broadcom, "Dragonfly: Western Energy Sector Targeted by Sophisticated Attack Group," *Symantec Enterprise Blogs/Threat Intelligence*, October 20, 2017, https://symantec-enterprise-blogs.security.com/blogs/threat-intelligence/dragonfly-energy-sector-cyber-attacks.

[2] Andy Greenburg, "Hackers Gain Direct Access to US Power Grid Controls," *WIRED*, September 6, 2017, https://www.wired.com/story/hackers-gain-switch-flipping-access-to-us-power-systems/.

[3] Kim Zetter, "Hackers Hijacked ASUS Software Updates to Install Backdoors on Thousands of Computers," *VICE*, March 25, 2019, https://www.vice.com/en_us/article/pan9wn/hackers-hijacked-asus-software-updates-to-install-backdoors-on-thousands-of-computers.

Executive Summary

Society has a software problem. 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. 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."[4]

With software come security flaws and a long tail of updates from vendors and maintainers. Unlike a physical system that is little modified once it has left the factory, software is subject to continual revision through updates and patches. This makes the supply for code long and subject to myriad flaws, both unintentional and malicious. The private sector's aggregated risk from software supply chain compromises continues to grow. Ever more feature-rich software is finding its way into a widening array of consumer products and enterprise services, enlarging the potential attack surface. Organizations increasingly outsource IT management and services to cloud computing and managed service providers (MSPs), raising the likelihood that a given firm will be impacted by an attack targeting one of these providers, like the successful penetration of eleven Saudi MSPs in 2018.[5] A similar kind of concentration is present in software development where firms can buy pre-built code from a third parties for complex or widely encountered tasks. Trek Networks, a US company, builds software to allow Internet of things (IoT) devices to communicate over the Internet. In 2020, it was informed of nineteen critical vulnerabilities in its products.[6] These vulnerabilities in one company's software impacted products from nearly a dozen other manufacturers, like Intel and Caterpillar, potentially affecting hundreds of millions of devices.

The public sector, particularly defense organizations, assumes even greater risk. A generation of Western defense systems, led by those in the United States, benefit from the advantages of Commercial Off-the-Shelf (COTS) procurement. Under a COTS model, defense organizations look to buy and repurpose or build from available commercial components to reduce cost, limit technological lag, and improve development speed. For the United States, COTS software has underpinned a generation of Department of Defense (DoD) systems, leveraging everything from miniaturized low-cost GPS receivers to high-bandwidth satellite data links with unmanned aerial vehicles (UAVs) and growing dependence on open-source software (OSS) in logistics and maintenance systems. A flaw in widely used software could undermine the DoD's ability to interpret and work with large quantities of sensor data. This attack could be innocuous and go undetected for months, like a 2017 incident in which malicious code was substituted for legitimate samples in the Python Package Manager.[7]

Software supply chain security remains an underappreciated domain of national security policymaking. The debate over 5G and telecommunications security, for example, has focused largely on hardware manufacturing and deployment, yet it is software in these devices that determines data's confidentiality and path through the global Internet. The push for open architecture in telecommunications, the Open Radio Access Network (ORAN) model, and industry trends toward network function virtualization mean even more hardware functionality will shift to software—making software supply chain security a critical competitive dimension. Exclusive focus on hardware security has resulted in missed opportunities for policy makers. Continued inaction to secure software supply chains risks compromising important intelligence, defense, and public policy programs and will undermine the long-term innovative potential of already faltering US technology dominance.

Software supply chain attacks are popular, they are impactful, and are used to great effect by states, especially China and Russia. High-profile attacks like NotPetya have forced policy makers to confront the importance of software supply chains, but only episodically and without leading

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

[5] Danny Palmer, "Cybersecurity: New Hacking Group Targets IT Companies in First Stage of Supply Chain Attacks," ZDNet, September 18, 2019, https://www.zdnet.com/article/cybersecurity-new-hacking-group-targets-it-companies-in-supply-chain-attack-campaign/.

[6] Sean Lyngaas, "'Ripple' Effect: Flaws Found in Protocols Impact Everything from Printers to Infusion Pumps," *CyberScoop, June 16, 2020,* https://www.cyberscoop.com/ripple-treck-vulnerabilities-jsof-forescout/.

[7] Dan Goodin, "Devs Unknowingly Use 'Malicious' Modules Snuck into Official Python Repository," Ars Technica, September 16, 2017, https://arstechnica.com/information-technology/2017/09/devs-unknowingly-use-malicious-modules-put-into-official-python-repository/.

to long-term security improvements.[8] Improved technical security measures could raise the cost of attacks, but the United States and its allies must respond to systemic threats and counter the efforts of states to undermine trust in software. The cost of these attacks is as much an issue for states in the European Union (EU) and Asia as it is for the United States. This report's first trend discusses patterns of state attacks against the software supply chain to motivate several recommendations on new alliance models and operational collaboration.

The security of the software supply chain matters as much as the codebase first delivered to a customer but receives comparably less attention and formal treatment in policy than secure development practices or vulnerability discovery. This report evaluates 115 software supply chain attacks and vulnerability disclosures collected from public reporting covering the past ten years. All of these incidents are based on public blogs, write-ups, and media articles so do not include private disclosures of attacks which never made it to the public domain. As such, even these trends likely undercount the frequency of attacks on certain high-value targets and are biased away from regions throughout the Global South that receive less focus from the cybersecurity industry. This report identifies five trends in software supply chain attacks over the past decade and offers three clusters of recommendations to address, mitigate, and counter them. The five trends identified are:

Deep Impact from State Actors: States have targeted software supply chains with great effect, alone constituting almost a quarter of this report's dataset.[9] The majority of cases surveyed here did, or could have, resulted in remote code execution. We found that states were more likely to hijack updates, a comparatively sophisticated distribution vector. Several of these cases, like the Dragonfly 2.0 attacks on energy sector targets, are representative of longer campaigns rather than single events. While there are examples from Egyptian, Indian, Iranian, North Korean, and Vietnamese actors, Russia and China were far and away the most frequent. **Examples:** CCleaner,[10] NotPetya,[11] Kingslayer,[12] SimDisk,[13] and ShadowPad.[14]

Hijacking Updates: These attacks were generally carried out by states or extremely capable actors. Updates that were signed either by stolen or forged certificates carried malware to targets. The advanced malware often contained components allowing it to spread further from the infected machine either along networks or in hardware. These attacks were more likely to encrypt data, target physical systems, or extract information, and, generally, were far more sophisticated than app store components. **Examples**: Flame,[15] Stuxnet,[16] CCleaner 1 and 2,[17] NotPetya,[18] Adobe pwdum7v71,[19] Webmin,[20] and PlugX.[21]

Undermining Code Signing: The technique relies on public key cryptography and a certificate system to ensure the integrity of updates and the identity of their authors. Overcoming its protections is a critical step in any supply chain attack, enabling anything from simple alterations of open-source code to complex nation-state espionage campaigns. It is the technical process that fosters trust in

[8] Dan Goodin, "Backdoor Built in to Widely Used Tax App Seeded Last Week's NotPetya Outbreak," Ars Technica, July 5, 2017, https://arstechnica.com/information-technology/2017/07/heavily-armed-police-raid-company-that-seeded-last-weeks-notpetya-outbreak/.

[9] Much like this dataset as a whole, the survey of state attacks is biased toward publicly reported incidents. Many such attacks and some disclosures have never reached the public domain and thus are not captured here.

[10] Lily Hay Newman, "Inside the Unnerving Supply Chain Attack That Corrupted CCleaner," *WIRED, April 17, 2018,* https://www.wired.com/story/inside-the-unnerving-supply-chain-attack-that-corrupted-ccleaner/.

[11] Goodin, "Backdoor."

[12] Howard Solomon, "Canadian Cyber Firm Confirms It Was the Victim Described in RSA Investigation," IT World Canada, February 23, 2017, https://www.itworldcanada.com/article/canadian-cyber-firm-confirms-it-was-the-victim-described-in-rsa-investigation/390903.

[13] Oscar Celestino Angelo Abendan II, "Trend Micro Investigates June 25 Cyber Attacks in South Korea," Trend Micro, July 1, 2013, https://www.trendmicro.com/vinfo/us/threat-encyclopedia/web-attack/124/trend-micro-investigates-june-25-cyber-attacks-in-south-korea.

[14] Kaspersky Lab, ShadowPad: How attackers hide backdoor in software used by hundreds of large companies around the world, press release, August 15, 2017, https://www.kaspersky.com/about/press-releases/2017_shadowpad-how-attackers-hide-backdoor-in-software-used-by-hundreds-of-large-companies-around-the-world.

[15] Swiat, "Flame Malware Collision Attack Explained," Microsoft Security Response Center, June 6, 2012, https://msrc-blog.microsoft.com/2012/06/06/flame-malware-collision-attack-explained; Alex Sotirov, "Analyzing the MD5 Collision in Flame," Speaker Deck, June 11, 2012, https://speakerdeck.com/asotirov/analyzing-the-md5-collision-in-flame?slide=7.

[16] Brandon Vigliarolo, "Stuxnet: The Smart Person's Guide," TechRepublic, August 15, 2017, https://www.techrepublic.com/article/stuxnet-the-smart-persons-guide.

[17] Newman, "Inside."

[18] Goodin, "Backdoor."

[19] Ryan Naraine, "Adobe Code Signing Infrastructure Hacked by 'Sophisticated Threat Actors,'" *Zero Day*, September 27, 2012**,** https://www.zdnet.com/article/adobe-code-signing-infrastructure-hacked-by-sophisticated-threat-actors.

[20] "Webmin 1.890 Exploit—What Happened?" Webmin, accessed July 15, 2020, http://www.webmin.com/exploit.html.

[21] Benson Sy, "PlugX Malware Found in Official Releases of League of Legends, Path of Exile," *Security Intelligence Blog*, January 19, 2015, https://blog.trendmicro.com/trendlabs-security-intelligence/plugx-malware-found-in-official-releases-of-league-of-legends-path-of-exile.

software and is the central protection against hijacked updates. **Examples:** ShadowHammer,[22] Naid/McRAT,[23] and BlackEnergy 3.[24]

Open-Source Compromise: These incidents saw attackers either modify open-source code by gaining account access or post their own packages with names similar to commonly used ones. The malicious code they spread usually stole victims' data and occasionally tried to target payment information. The actors were usually criminals, and their attacks were generally quickly discovered. **Examples:** Cdorked/Darkleech,[25] RubyGems Backdoor,[26] HackTask,[27] Colourama,[28] JavaScript 2018 Backdoor,[29] and 2017 PyPI repository attack.[30]

App Store Attacks: These attacks used the Google Play Store, Apple's App Store, and other third-party app distributors to spread malware to mobile devices. Usually, the apps were designed by attackers to appear legitimate, though some were legitimate apps that they managed to compromise. The malicious apps tended to run adware, steal payment information, and extract data sent to a server operated by the attackers. Most perpetrators were criminals, though some state-backed tracking also occurred. **Examples:** Sandworm's Android attack,[31] ExpensiveWall,[32] BankBot,[33] Gooligan,[34] and XcodeGhost.[35]

These trends show software supply chain attacks are popular and impactful. They exploit natural seams between organizations and abuse relationships where users expect to find trustworthy code. These attacks are also impactful—targeting the supply chain for code can help magnify the value of a breach and sow distrust in widely used open-source projects. Second, these attacks can drive compromise deep into organization's technology stack, undermining development and administrative tools, code-signing, and device firmware. And, third, software supply chain attacks have strategic utility for state actors and have been used to great effect, especially by Russian and Chinese groups. This trend is likely to continue and should motivate action from US policy makers.

The implication for national security policymakers and the cybersecurity community across the US and allies is that change is necessary to raise the cost, and lower the impact, of software supply chain attacks. New efforts should use existing security controls and make them accessible and low cost for developers. Policymakers should drive new resources and support to open source projects and enable better software supply chain security. Civil society and the cybersecurity standards community hold great potential to help sustain these changes once initiated, especially for open source. The US and allies need to pursue new approaches to joint activity across the Atlantic and Pacific to counter state threats and facilitate more effective long-term investigations of criminal actors. Traditional alliance structures may be insufficient to address threats to major industry players and software supply chains relied on by intelligence and defense departments and agencies.

This report offers three clusters of recommendations to policy makers and industry to address the insecurity of the software supply chain. **First, improve the baseline**. The lynchpin of any effort to improve the security of software

[22] Zetter, "Hackers."

[23] Liam Tung, "Java Zero-Day Malware 'Was Signed with Certificates Stolen from Security Vendor,'" ZDNet, March 4, 2013, https://www.zdnet.com/article/java-zero-day-malware-was-signed-with-certificates-stolen-from-security-vendor.

[24] Robert M. Lee, Michael J. Assante, and Tim Conway, *Analysis of the Cyber Attack on the Ukrainian Power Grid, SANS ICS and E-ISAC, March 18, 2016,* https://ics.sans.org/media/E-ISAC_SANS_Ukraine_DUC_5.pdf.

[25] Neil McAllister, "Stealthy, Malware-Spewing Server Attack Not Limited to Apache," *Register, May 8, 2013,* https://www.theregister.com/2013/05/08/cdorked_latest_details/.

[26] Jeff Smykil, "MobileMe Proving to Be Valuable to Theft Victims," *Ars Technica, September 2, 2009,* https://arstechnica.com/gadgets/2009/09/mobilemes-value-greater-than-meets-the-eye-for-some/#issuecomment-522967049.

[27] Thomas Claburn, "This Typosquatting Attack on NPM Went Undetected for 2 Weeks," *Register, August 2, 2017,* https://www.theregister.com/2017/08/02/typosquatting_npm/.

[28] Dan Goodin, "Two New Supply-Chain Attacks Come to Light in Less Than a Week," *Ars Technica, October 23, 2018,* https://arstechnica.com/information-technology/2018/10/two-new-supply-chain-attacks-come-to-light-in-less-than-a-week/.

[29] Catalin Cimpanu, "Hacker Backdoors Popular JavaScript Library to Steal Bitcoin Funds," *Zero Day, November 26, 2018,* https://www.zdnet.com/article/hacker-backdoors-popular-javascript-library-to-steal-bitcoin-funds/.

[30] Goodin, "Two."

[31] Andy Greenberg, "Russia's 'Sandworm' Hackers Also Targeted Android Phones," *WIRED, November 21, 2019,* https://www.wired.com/story/sandworm-android-malware/.

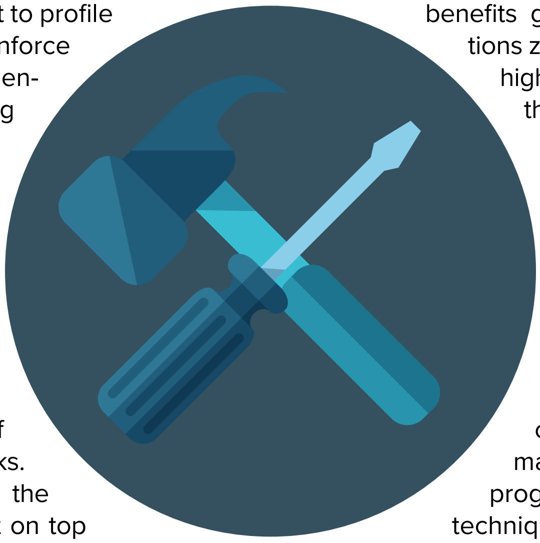
[32] Check Point, "ExpensiveWall: A Dangerous 'Packed' Malware on Google Play That Will Hit Your Wallet," *Check Point Blog, accessed June 15, 2020,* https://blog.checkpoint.com/2017/09/14/expensivewall-dangerous-packed-malware-google-play-will-hit-wallet/.

[33] Danny Palmer, "BankBot Android Malware Sneaks into the Google Play Store—for the Third Time," *ZDNet, November 9, 2017,* https://www.zdnet.com/article/bankbot-android-malware-sneaks-into-the-google-play-store-for-the-third-time/.

[34] Check Point, "More Than 1 Million Google Accounts Breached by Gooligan," *Check Point Blog, accessed July 15, 2020,* https://blog.checkpoint.com/2016/11/30/1-million-google-accounts-breached-gooligan/.

[35] 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/.

supply chains broadly will be what impacts the largest number of codebases, not what improves one codebase the most. Perhaps the most useful thing the policy community can do is offer support for widely compatible standards and tools to reduce the burden of secure software supply chain management on developers and project owners. Best practices are not effective in isolation, and below a certain threshold it is difficult to profile which vendors and project owners enforce good security practices. The recommendations in this report focus on bringing the best of public and private sector supply chain security tools into the public domain, aligning these tools with widely supported standards, and calling out key stakeholders to help share both how to adopt and assess against them.



Second, better protect open source. Open-source code forms the basis of most enterprise systems and networks. Even large proprietary projects, like the Windows operating system, are built on top of huge quantities of open-source code. The security of open-source projects, and the apparent ease with which attackers can introduce insecure code, is a continuing concern. The fluidity with which anyone can commit code to an open-source project is at once a core strength and glaring weakness. The policy community must support efforts to secure open-source projects, or it will watch a critical and innovative ecosystem wither.

Third, counter systemic threats. Trust is the critical coin of the realm and the United States must work with international partners to protect against deliberate efforts to undermine software supply chains. Efforts by states to impersonate software vendors undermines defender's ability to patch flaws in code and improve the security of software through the entirety of its lifecycle. This lifecycle is critical to sustaining the national security advantages granted by current and near-future technologies like sensor fusion networks, autonomous supply chains, and "smart" devices. Failure to protect the ability to trust software could cripple the benefits gained from it. These recommendations zero in on systemic threats to trust and highlight new institutional responses from the US and allies to mitigate the activities of states like Russia and China. Transoceanic responses are needed to protect the software supply chain and the US can extend existing partnerships to support more effective joint attribution and collaboration with allies.

Compromising the software supply chain enables attackers to deliver malicious code in the guise of trusted programs. This can be a terribly effective technique to spread an attack widely and in targeting well-secured systems. These risks are particularly acute for the national security community whose ability to churn through huge quantities of surveillance data, run complex weapon systems, and support modern logistics systems is dependent on software, most of it developed outside of government. The solution is not panic nor is it a moonshot, but rather renewed focus on software supply chain security practices, new investment from public and private sectors, and revisions to public policy that emphasize raising the lowest common denominator of security behavior while countering the most impactful attacks. For more on this project and the dataset behind it, please visit us on the web here.

1. Introduction

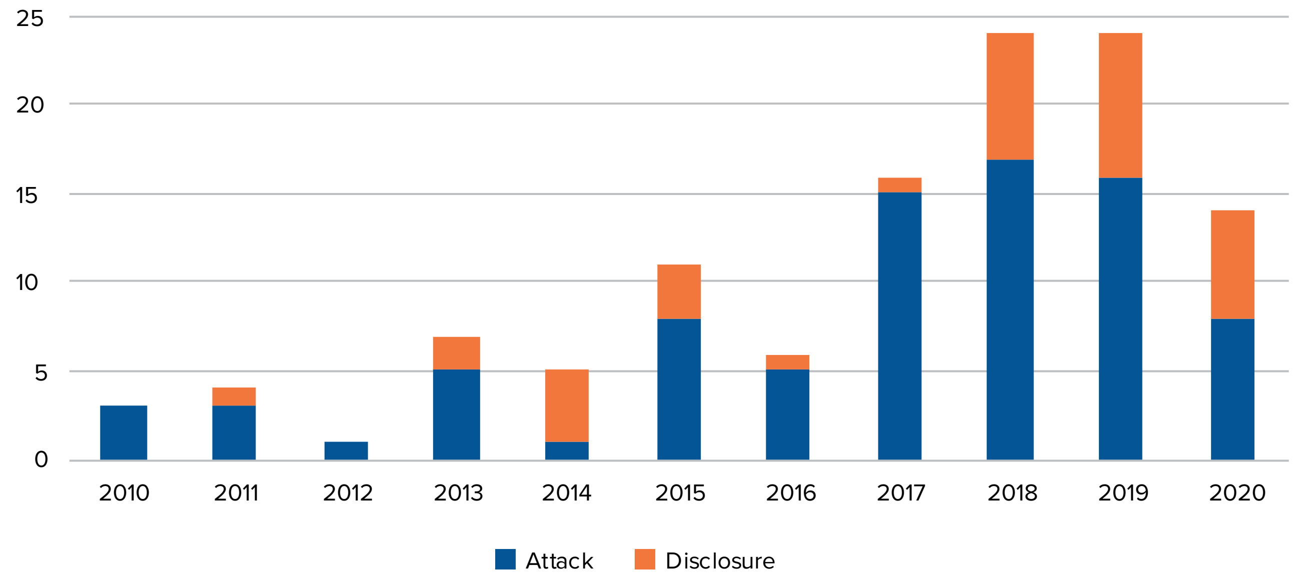
The state of security in the software supply chain is inadequate and, in some critical respects, getting worse. The policy community must refocus on this topic amidst competing national security priorities and do more to incentivize and support private sector security. Failure to do so creates new and systemic national security risks for the United States and its allies. Attackers capitalizing on vulnerable software supply chains are able to compromise trusted software and important cybersecurity protections to impact large numbers of critical systems.

Significant economic output depends on the security of software every day—the sudden shift to remote work is exemplary of this dependence. Without reliable video conferencing, email, and file sharing, much of the world's knowledge work would slow to a crawl. Even intangible capacities, like our capacity to innovate, rely in large part on digital tools to collaborate, coordinate, and revise. All of these tasks are dependent on software. Critical defense systems are equally dependent on software. Digitized logistics processes, the ability to work with massive quantities of data, semiautonomous sensors, and munitions all depend on a chain of digital logic embedded in software programs. Part of the F-35's own software supply chain, the Autonomous Logistics Information System (ALIS), is plagued with vulnerabilities and testing gaps, potentially compromising the entire weapons platform.[36] The supply chain for these physical products is a recognized priority, but the software enabling these systems must be as well.

Software supply chain attacks are not an esoteric or isolated tactic, as this report and its associated dataset shows; they are popular, impactful, and have been used to great effect by state actors. In this dataset, we have collected 115 instances, going back a decade, of publicly reported attacks on the software supply chain or disclosure of high-impact vulnerabilities likely to be exploited in such attacks.[37] Each instance of an attack or vulnerability was coded with incident name, date, victim, and likely source, while categorizing the basic technical characteristics of each. Figure 1 shows the distribution of attacks and disclosures in the dataset in the examined period.

A software supply chain vulnerability is any software vulnerability that can evolve into an attack, if exploited. In our

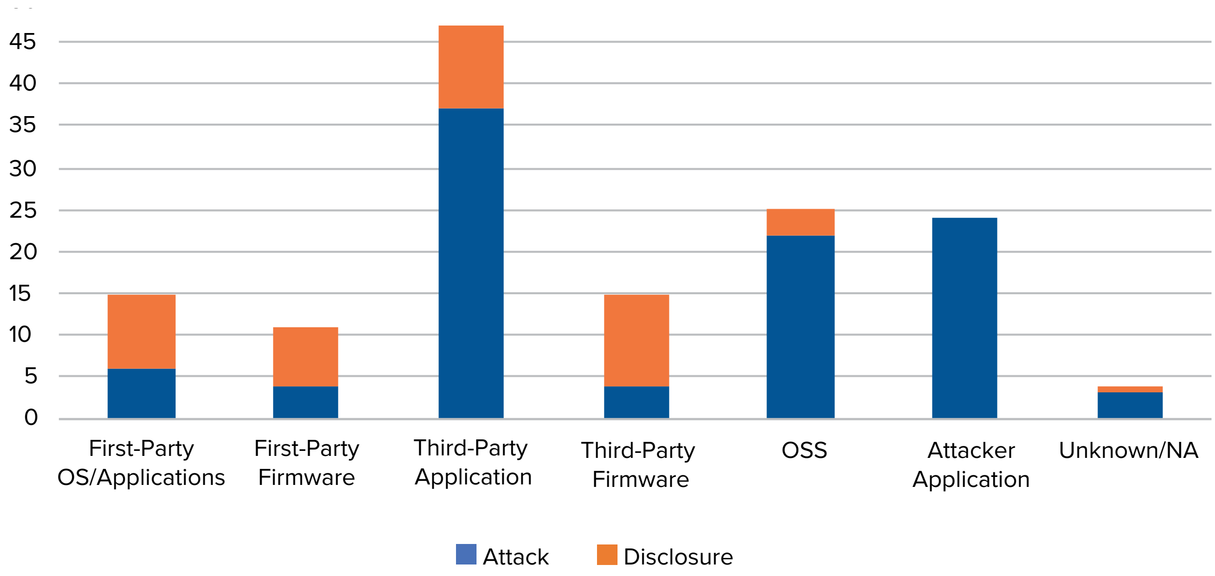
Figure 1. Changes in Attacks and Disclosures Over the Years



[36] Dan Grazier, "Uncorrected Design Flaws, Cyber-Vulnerabilities, and Unreliability Plague the F-35 Program," POGO, March 24, 2020, https://www.pogo.org/analysis/2020/03/uncorrected-design-flaws-cyber-vulnerabilities-and-unreliability-plague-the-f-35-program/.

[37] All of the figures and charts in this report reflect our dataset, which is a consistent but not random sample of the total population of software supply chain attacks and vulnerabilities going back a decade. We draw no conclusions as to the frequency of attacks or vulnerabilities for this population, only trends within this dataset, and can make no claims of statistical significance as a result.

Figure 2. Codebase by Attack and Disclosure



dataset, we include vulnerabilities that would enable the injection or distribution of malicious code rather than what would be found in the code's payload, the means of access rather than the harmful payloads. Figure 2 shows the distribution of attacks versus disclosures in the dataset across impacted codebases.

These incidents reveal a stunning diversity—there are as many potential target types in the supply chain as there are types of software. Certain categories have consistently been in the crosshairs for years now: software updates, firmware, and third-party apps. In this dataset, others are being targeted more today than in the past. For instance, as smartphones become ubiquitous, mobile apps, both attacker-made and attacker-infiltrated, have become increasingly popular targets—as have open-source libraries on which much software relies.

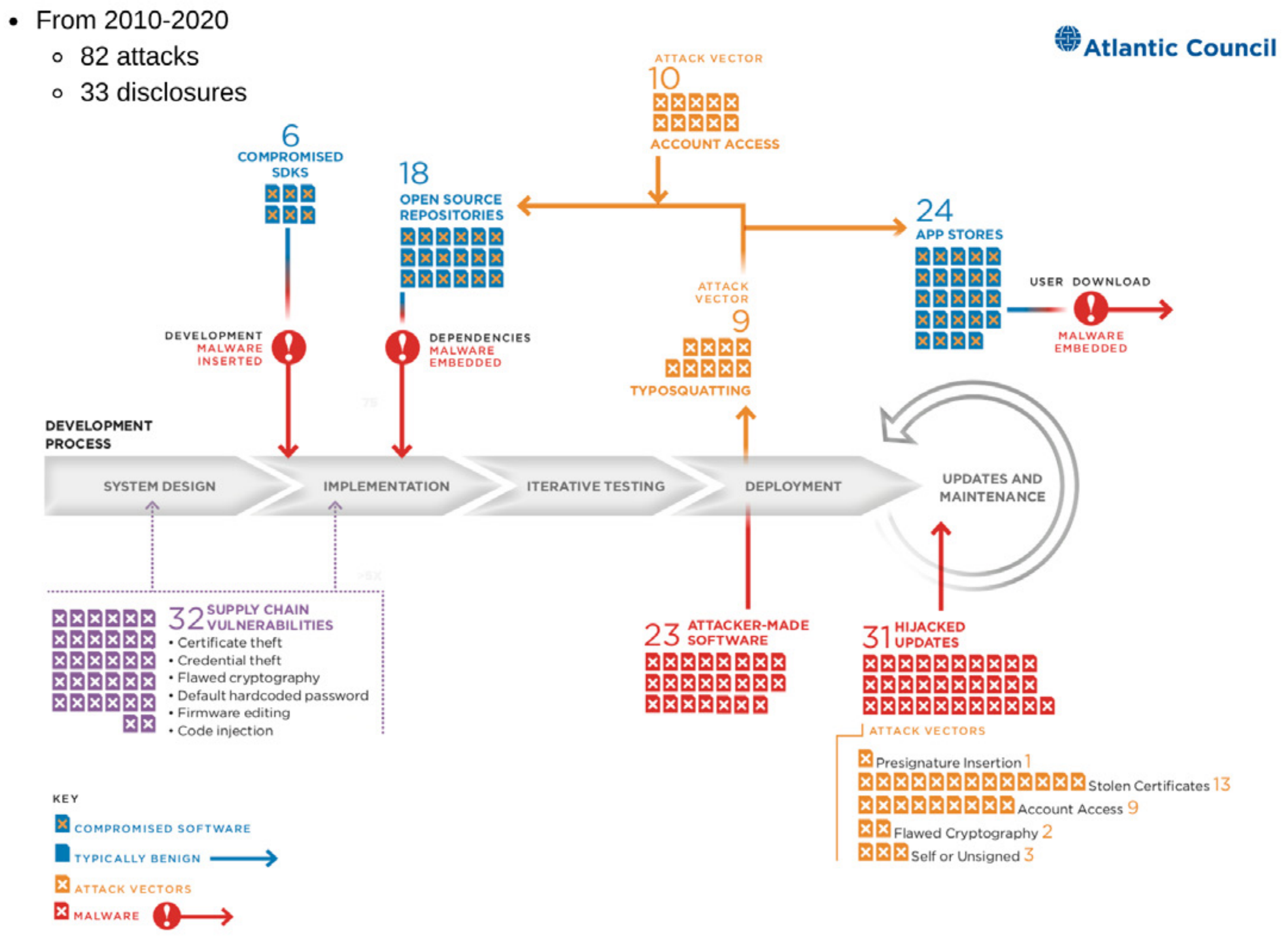
Figure 3 uses an abstracted model to show the distribution of the 115 attacks in our dataset and disclosures across a notional software supply chain. There is no good way to concisely represent software development and this graphic is not intended to capture all of the intricacies of the process. This representation of a waterfall-style model matches with much of the software captured in the study. Agile development methodologies like software Development/IT Operations (DevOps) and Development/Security/Operations (DevSecOps) are important but are also no magic solution for software supply chain security issues. These Agile approaches and the philosophy of continuous integration create new opportunities to quickly propagate risk through a codebase.

While there are attacks through this supply chain model, the figure highlights their concerning concentration against app stores, update processes, OSS projects, and around code signing. Incidents within these types often shared attack vectors, codebases, and distribution vectors, and were carried out by similar actors with similar goals. Figure 3 underlines just how many different opportunities exist to undermine a software supply chain.

The purpose of this dataset is to compile a variety of software supply chain attacks and discovered vulnerabilities, and to catalogue the different characteristics of each incident. While the dataset is not exhaustive, we took care to use consistent definitions and typologies throughout. We defined attacks as occurring when attackers access and edit software somewhere in a software development supply chain, compromising a target farther up on the chain, by inserting their own malicious code. Disclosures that came alongside an ongoing attack were coded as an attack. Attacks don't require any public disclosure of a vulnerability, only public notice of the attack itself. Software supply chain vulnerabilities are any software vulnerability that, if exploited, would comprise an attack. This flexibility includes a wide variety of critical vulnerabilities, so we limited their scope to those that would enable injection of malicious code, not just compound the effects of an attack's payload.

All incidents were collected via manual online search for media articles, industry reporting, and researcher write-ups of incidents, using information in the public domain. As a result, the dataset here is a lower bound on the population

Figure 3. Software Supply Chain Life Cycle



of attacks which undercount incidents that have never been made public and does not include vulnerability disclosures that were not explicitly and publicly reported, i.e., does not include information from patch notes (though this is an opportunity for future work). To analyze trends in these attacks and disclosures, we established five key measures to better understand the different elements of supply chain compromise: attack vector, distribution vector, affected codebase, impact, and supply chain potential. None of these five measures is exclusive: a single entry in the database can have multiple insertion points or methods, distribution vectors, and so on. Information on an entry's date, responsible actor, downstream target, links to relevant articles, and a paragraph of technical summary was also included. More detailed descriptions of the dataset's fields can be found in the codebook online.[38]

Although supply chain security has become a hot-button issue for both private industry and policy makers in recent years, the problem is not being addressed holistically and software has largely taken a back seat to 5G in public debate. There are clear and critical policy gaps that exist in securing our software supply chains. Users are increasingly vulnerable, whether it be as a result of malicious apps in app stores, forged digital certificates, or vulnerable open-source code frameworks.

Moreover, this issue is not limited to the United States. State attackers have regularly employed software supply chain intrusions to attack the United States and its allies. Governments around the world are increasingly dependent on open-source and commercial off-the-shelf software for systems relevant to national security, and are exposed to

[38] *Breaking Trust: Shades of Crisis Across an Insecure Software Supply Chain*, Atlantic Council, accessed July 15, 2020, https://www.atlanticcouncil.org/breaking-trust/.

Coding the Dataset

Date: Best estimated start date of the attack. When no start date is identifiable, discovery date is used instead.

Name: Name(s) of the attack/incident. Multiple names are included where possible.

Software Supply Chain Attack: A software supply chain attack occurs when an attacker accesses and edits software in the complex software development supply chain to compromise a target farther up on the chain by inserting their own malicious code.

Software Supply Chain Vulnerability: A software supply chain vulnerability is any software vulnerability that could evolve into an attack, if exploited (but which has not been used in an attack).

Affected Code: The code modified by attackers, or the code that had a vulnerability in it.

Code Owner: Who owned that code, or, if open source, the repository name.

Codebase: Categories describing the codebase, product, or service modified by attackers.
First party refers to the same author as the creator of the machine; third party is code written by another author whose source code is not publicly available; and open source is code that is publicly viewable.

Attack Vector: How the attacker was able to edit the affected code without detection.

Distribution Vector: How the attacker was able to distribute the modified code.

vulnerabilities similar to those faced by the United States. Attacks such as PhantomLance[39] and KingSlayer[40] demonstrate that an attack on systems in one country can easily spread to other states through overlapping supply chains and shared software suppliers. The global nature of software supply chains and the international character of threats make it imperative for the United States to work with partners in the private sector and allies around the world to address this security challenge.

This report offers a series of recommendations that call for a coalition response to the systemic threats posed by software supply chain attacks and vulnerabilities. Working within existing alliance structures while broadening cooperation will allow the United States and its allies to limit the threat actors seeking to undermine trust in the global technology ecosystem, while adapting current alliance networks to the changing threat landscape. Shifting global cybersecurity norms, increasing information sharing to support operational collaboration, and reshaping the focus of interagency equities are all crucial for addressing the serious national security implications of attacks against the software supply chain.

There are opportunities for improvement and a closing window in which to seize them. The remainder of this report discusses five trends in the attacks and disclosures surveyed, including the damaging use of software supply chain attacks by a handful of major adversaries of the United States, efforts to undermine code-signing processes and hijack software updates, and the popularity of attacks on open-source projects and app stores. Following this is a summary of key takeaways from the dataset and specific policy recommendations to drive improved baseline security across software supply chains, better protect open-source software, and counter systemic threats to these supply chains.

[39] Abeerah Hashim, "PhantomLance Malware Campaign Has Taken Over Android Play Store," Latest Hacking News, May 4, 2020, https://latesthackingnews.com/2020/05/04/phantomlance-malware-campaign-has-taken-over-android-play-store/.

[40] Howard Solomon, "Canadian Cyber Firm Confirms It Was the Victim Described in RSA Investigation," IT World Canada, February 23, 2017, https://www.itworldcanada.com/article/canadian-cyber-firm-confirms-it-was-the-victim-described-in-rsa-investigation/390903.

2. Attacks on the Software Supply Chain

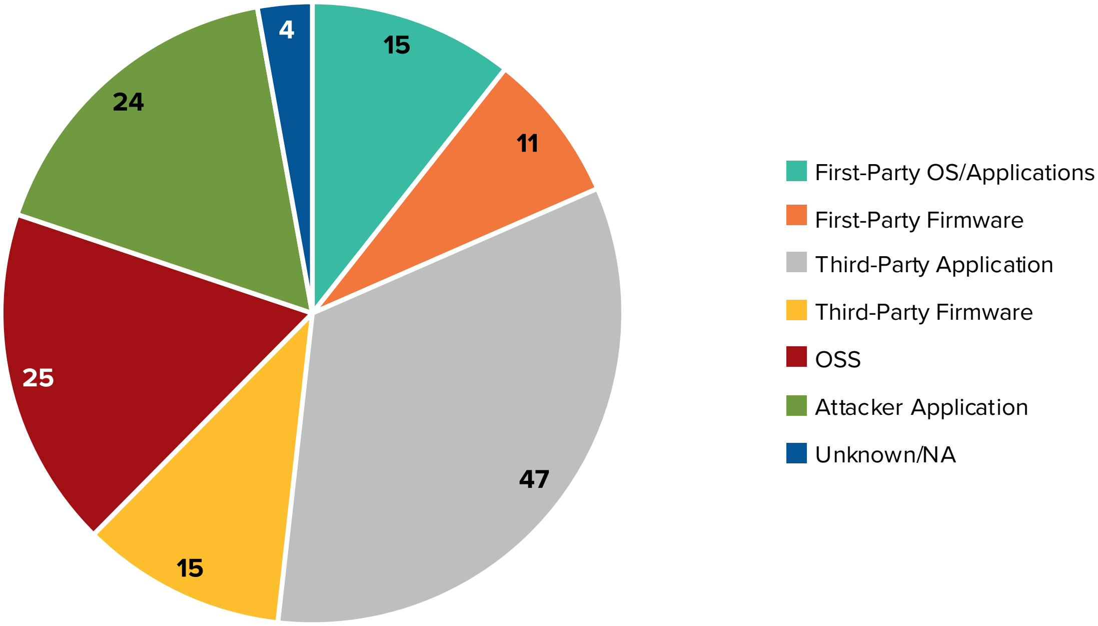
A software supply chain attack occurs when an attacker accesses and modifies software in the complex software development supply chain to compromise a target farther down on the chain by inserting their own malicious code. These inserts can be used to further modify code by obtaining system permissions or to directly deliver a malicious payload. 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. There are efforts underway to make tracking these dependencies, and discovering them in the event of an incident, more straightforward. The most widely recognized is the Software Bill of Materials (SBOM) multi-stakeholder initiative coordinated by the US Department of Commerce, which remains in development.

Software supply chain attacks take advantage of established channels of system verification to gain privileged access to systems and to compromise large networks. They undermine foundational tenets of trust in software development. Even with the increased exposure gained by these high-profile attacks, we are only just beginning to understand how wide-reaching the impact can be.

The code that attackers and researchers target has changed over time within this dataset. Figure 4 shows the distribution of codebases targeted by these incidents, while Figure 5 shows that distribution over time. Mobile apps have grown more frequent in the dataset, especially those created by attackers and marketed as legitimate. For instance, in 2017, the Android app Lovely Wallpaper hid malware under the guise of providing phone background images.[41] The malware would gain device permissions and charge users' accounts for "premium" services they had not signed up for. It, and at least fifty other apps hiding the same payload, infected as many as 4.2 million devices, and successors continued to infiltrate the Google Play Store after the original offenders were removed.

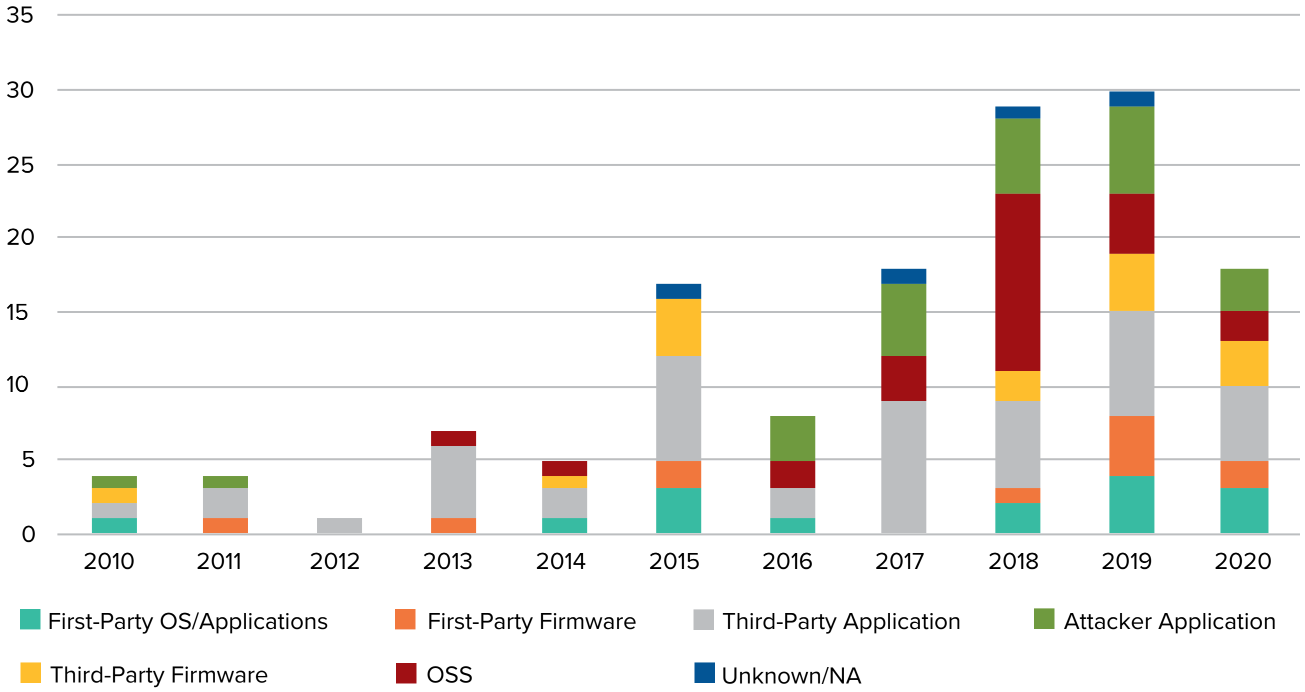
Third-party firmware is also an increasingly popular target for attackers and researchers alike—a particularly troublesome development given the inherent difficulty of patching firmware and its ability to make privileged edits and

Figure 4. Codebases Targeted



[41] Check Point, "ExpensiveWall: A Dangerous 'Packed' Malware on Google Play That Will Hit Your Wallet," *Check Point Blog, accessed June 15, 2020,* https://blog.checkpoint.com/2017/09/14/expensivewall-dangerous-packed-malware-google-play-will-hit-wallet/.

Figure 5. Changes in Code Targeted Over the Years



avoid much anti-malware program detection. The Equation Group demonstrated the potency of these attacks with its GrayFish attack on hard drive firmware, which allowed it to record system data in a nearly inaccessible portion of machine memory for remote extraction at a later point.[42] Removing the data cache was nearly impossible short of destroying the physical system, and even detecting the infection was generally infeasible.

Proprietary firmware, too, is more frequently discovered to be insecure. In August 2019, McAfee researchers uncovered a vulnerability in Avaya firmware for the 9600 series desk phone, which could have been present at 90 percent of Fortune 500 companies.[43] The bug resulted from the inclusion of unmaintained open-source code and would have allowed attackers to crash a system, run code with root access, and record or listen in on the phone's audio.

From these general patterns across the 115 incidents in this dataset we observed five more specific trends:

1. *Deep Impact:* State actors target the software supply chain and do so to great effect.

2. *Abusing Trust:* Code signing is a deeply impactful tactic for compromising software supply chains as it can be used to undermine other related security schemes, including program attestation.

3. *Breaking the Chain:* Hijacked updates are a common and impactful means of compromising supply chains and they recurred throughout the decade despite being a well-recognized attack vector.

4. *Poisoning the Well*: Attacks on OSS were poplar, but unnervingly simple in many cases.

5. *Downloading Trouble*: App stores represent a poorly addressed source of risk to mobile device users as they remain popular despite years of evidence of security lapses.

[42] Kim Zetter, "How the NSA's Firmware Hacking Works and Why It's So Unsettling," *WIRED, February 22, 2015,* https://www.wired.com/2015/02/nsa-firmware-hacking/.

[43] Charlie Osborne, "Decade-Old Remote Code Execution Bug Found in Phones Used by Fortune 500," *Zero Day, August 8, 2019,* https://www.zdnet.com/article/decade-old-remote-code-execution-bug-found-in-phone-used-by-up-to-90-percent-of-fortune-500/.

2.1 Deep Impact: States and Software Supply Chain Attacks

States have used software supply chain attacks to great effect. Hijacked updates have routinely delivered the most crippling state-backed attacks, thanks in part to a continued failure to secure the code-signing process. 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[44] and the Equifax data breach in 2017,[45] impacted millions of users, showcasing the immense potential scale of software supply chain attacks and their strategic utility for states. For Russia, these attacks have meant access to foreign critical infrastructure, while for China, they have facilitated a massive and multifaceted espionage effort. This section discusses trends in known state software supply chain attacks supported by publicly reported attribution, focused on four actors: Russia, China, Iran, and North Korea. The data in this report also include incidents linked to Egypt, India, the United States, and Vietnam, for a total of 27 distinct attacks.

Russia

Russian actors were responsible for the 2017 NotPetya attack,[46] one of the most destructive software supply chain attacks to date. Four of the five attacks attributed to Russia in our dataset involved initial insertion of malicious code into a third-party app, made possible by gaining access to an account with editing permissions. After inserting a malicious payload, Russia relied on diverse means to distribute this code, including hijacked updates, a worm component, phishing, and a hardware component. Notably, every attack involved multiple vectors for distribution.

Interestingly, three of the five attacks involved downstream targets in the energy sector. For instance, the 2015 Dragonfly 2.0 attack relied on a variety of vectors—including spear-phishing, watering-hole-style attacks, and Trojanized software updates—to obtain network credentials from targets in the US, Swiss, and Turkish energy sectors.[47] Launched by Russian APT Energetic Bear, attackers were able to gain operational control of interfaces that power company engineers use to send commands to equipment like circuit breakers, giving them the ability to stop the flow of electricity into homes and businesses in the United States. Russian attackers similarly compromised Ukraine's power grid in 2015, interrupting service to 225,000 customers in a complex infrastructure attack that involved spear phishing, credential stealing, malware insertion, distributed denial-of-service (DDoS) attacks on call centers, and firmware attacks.[48] Russia has repeatedly engaged in software supply chain attacks against Ukrainian entities or programs since 2015, in line with its practice of testing cyber war tactics in Ukraine.

China

Of the state actors featured in the dataset, China has conducted the most software supply chain attacks (eleven) and demonstrated the greatest level of consistency in attack and distribution methods. The earliest case attributed to Chinese actors was in 2011, suggesting that China has been targeting software supply chains earlier than other state actors. Most Chinese attacks relied on a third-party application for their initial insertion point; in the majority of cases, the affected code was found in an update server. Chinese attacks were notably consistent in the method of distribution: eight of eleven cases relied on hijacked updates to distribute malicious code, while several cases relied on supply chain service providers. For instance, in the 2017 Kingslayer attack, Chinese attackers (likely APT31) targeted a Windows IT admin application to include malicious code under a valid signature, which could spread either by updating or downloading the application.[49] The attack compromised a huge list of higher-education institutions, military organizations, governments, banks, IT and telecom providers, and other enterprises, as the malware installed a secondary package that could up- and download files, execute programs, and run arbitrary shell commands. With respect to impact, Chinese attacks tended to be of vast scale, gaining access to the personal data of millions of users, or impacting hundreds of companies.

Chinese software supply attacks are aimed more at corporate entities; eight attacks had companies and dependent users as their downstream targets. Given that all Chinese attacks resulted (or could have resulted) in data extraction, this data is consistent with continuing US concerns about Chinese intellectual property theft and economic espionage. The 2020 GoldenSpy malware notably targeted a

[44] Goodin, "Backdoor."

[45] Steven J. Vaughan-Nichols, "Equifax Blames Open-Source Software for its Record-Breaking Security Breach: Report," *ZDNet, September 11, 2017,* https://www.zdnet.com/article/equifax-blames-open-source-software-for-its-record-breaking-security-breach/.

[46] Goodin, "Backdoor."

[47] Sean Michael Kerner, "Dragonfly 2.0 Hackers Targeting the Energy Sector, Symantec Finds," *eWeek, September 6, 2017,* https://www.eweek.com/security/dragonfly-2.0-hackers-targeting-the-energy-sector-symantec-finds.

[48] Lee, Assante, and Conway, *Analysis.*

[49] Solomon, "Canadian."

multinational tech vendor servicing Western defense sectors by requiring it to install tax-paying software embedded with sophisticated malware while operating in China. We found that the greatest number of attacks occurred in 2017.[50] The timing may have been influenced by the conflicting, but often hostile, moves taken against China by US President Donald J. Trump's administration in its first year in office.

North Korea

The dataset features one software supply chain attack likely linked to North Korea—the 2013 compromise of two file-sharing services' auto-update features in order to launch a DDoS attack on South Korean government websites.[51] The attack was launched by the cyber adversary responsible for the "DarkSeoul" attack on South Korean banking and media in March 2013 and is consistent with other attacks by the group in the past.[52] This group has previously launched cyberattacks on days of historical significance in the United States and South Korea. The attack featured in the dataset occurred on June 25, the anniversary of North Korea's invasion of South Korea in 1950, which marked the start of the Korean War. In early 2013, North Korea also conducted an underground nuclear missile test and, in response to tightening sanctions by the United Nations in March, cut off communications with South Korea, threatened to launch a preemptive nuclear strike against the United States and South Korea, and pledged to restart its Yongbyon nuclear plant to provide material for its weapons program. The high-profile nature of the software attack on June 25 and the extensive damage it caused may accordingly be seen in the context of North Korea's escalation of military provocations at the time.

Technical elements of the attack are also indicative of methods attributed to the adversary and North Korea more generally. The attack targeted a third-party app and was distributed through a hijacked software update. The dropped malware also allowed for remote code execution and established a botnet to carry out a DDoS attack, consistent with North Korea's history of launching DDoS attacks (e.g., by North Korean APT Hidden Cobra). It is important to note that the adversary's targeting of IP addresses that serve as DNS name servers demonstrates careful research into the target in order to maximize damage, an approach also seen in the March 2013 "DarkSeoul" attack.

Iran

The dataset features one software supply chain attack weakly attributed to Iran—the 2020 Kwampirs malware campaign targeting companies in the industrial control systems sector, especially the energy industry.[53] The attack initially targeted supply chain software providers through exploitation of default passwords. The attackers distributed the Kwampirs Remote Access Trojan (RAT) through supply chain software vendors who would install infected devices on the target's network. The attack resulted, or likely resulted, in backdoor access and data extraction. The Kwampirs RAT was previously deployed in 2018 by a group called Orangeworm in similar attacks against supply chain software companies in the healthcare sector, and different entities feeding into the healthcare supply chain. While these attacks and the current campaign have not been directly attributed to Iran, the Kwampirs malware was found to contain numerous similarities to the Shamoon data-wiping malware used by Iranian-linked APT33, and also employed in multiple attacks against the energy sector. FireEye, a California-based cybersecurity firm, noted that targeting of organizations involved in energy and petrochemicals reflects a common interest and continuing thread among suspected Iranian threat groups. Previous reports suggest Iranian actors have targeted energy sector organizations headquartered in the United States, Saudi Arabia, and South Korea in an attempt to gain insight into regional rivals and potential competitors.

Other State Attacks

Other state actors have also engaged in software supply chain attacks; this dataset features incidents attributed to the United States, India, Egypt, and Vietnam. Stuxnet[54]—widely attributed to the United States and Israel—was one of the earliest examples to use stolen certificates and potentially one of a handful to leverage a hardware vector to compromise. Two campaigns were attributed to the US-attributed Equation Group (PassFreely[55] and EquationDrug

[50] Ken Dilanian, "Spyware Hidden in Chinese Tax Software Was Probably Planted by a Nation-State, Say Experts," NBC News, June 25, 2020, https://www.nbcnews.com/tech/security/spyware-hidden-chinese-tax-software-was-probably-planted-nation-state-n1231975.

[51] Abendan, "Trend Micro."

[52] Choe Sang-Hun, "Computer Networks in South Korea Are Paralyzed in Cyberattacks," *New York Times, March 20, 2013,* https://www.nytimes.com/2013/03/21/world/asia/south-korea-computer-network-crashes.html.

[53] Catalin Cimpanu, "FBI Warns About Ongoing Attacks Against Software Supply Chain Companies," *Zero Day, February 10, 2020,* https://www.zdnet.com/article/fbi-warns-about-ongoing-attacks-against-software-supply-chain-companies/.

[54] Michael Joseph Gross, "A Declaration of Cyber-War," *Vanity Fair, March 2, 2011,* https://www.vanityfair.com/news/2011/03/stuxnet-201104.

[55] Matthew J. Schwartz, "PassFreely Attack Bypasses Oracle Database Authentication," Bank Info Security, April 26, 2017, https://www.bankinfosecurity.com/passfreely-attack-bypasses-oracle-database-authentication-a-9868.

& GrayFish[56]). In 2013, PassFreely enabled bypass of the authentication process of Oracle Database servers and access to SWIFT (Society for Worldwide Interbank Financial Telecommunication) money transfer authentication. In 2015, the EquationDrug & GrayFish programs used versions of nls_933w.dll to communicate with a C&C server to flash malicious copies of firmware onto a device's hard disk drive, infecting 500 devices. Both attacks involved a supply chain service provider as a distribution vector.

The United States is generally distinguished from several of the other states highlighted here by formulation and operation of due process constraints on how attacks like these are developed, targeted, and deployed. One former senior US national security official reflected of Stuxnet, "If a government were going to do something like this, a responsible government, then it would have to go through a bureaucracy, a clearance process… It just says lawyers all over it".[57] The operational constraints imposed by democratic accountability and the trend toward what Peter Berkowitz labeled the "lawyering of war"[58] offer some meaningful distance between the United States and several of the other states on this list, though those distinctions blur where legal protections are ignored or interpreted beyond the bounds of accepted logic.[59]

The attacks attributed to the other three states notably all involved attacker applications uploaded to a proprietary app store, leading to data extraction and often remote code execution. A January 2020 attack involving three malicious apps on Google Play Store was linked to APT SideWinder (previously attributed to India). The attack exploited a serious zero-day vulnerability through which the CallCam, Camero, and File CryptManager apps, once downloaded, could extract extensive machine data, including screenshots, and send them to a C&C server. The Egyptian government is believed to have been involved in a hacking campaign against Egyptian human rights activists, in which attackers crafted applications that used OAuth Phishing, generating fake requests for access to Google, Yahoo, Hotmail, and Outlook accounts before stealing emails. The attack also led to 5,000 downloads of malicious mobile apps from the Google Play Store that would enable attackers to view call log information. Finally attackers connected to Vietnam-linked APT 32 (OceanLotus), used a variety of techniques to sneak malware into various browser cleanup apps on the Google Play Store; the attack sent device specs to a C&C server, allowing the attackers to download custom-designed malware onto target devices, likely for espionage. Three attacks from 2011 and 2012 (Duqu, Flame, Adobe code signing hack) have not been attributed to a particular state, but likely involved state actors based on the highly sophisticated and targeted nature of the attacks. These attacks relied on a variety of attack and distribution vectors, and were some of the most destructive attacks in the dataset.

As discussed in the introduction, this report's findings constitute a lower bound on the total population of supply chain attacks and disclosures. The dataset's contents are shaped by the limitations of publicly available research on software supply chain security—both in the scope and geographical focus of research efforts. Accordingly, the dataset focuses on incidents involving state actors traditionally covered in cybersecurity research, namely, China and Russia, North Korea, and Iran while understating those from countries in the developing world and Global South.

2.2 Abusing Trust: Code Signing

Code-signing issues were among the most prolific attack vectors in our analysis, with many attacks stemming from self-signed certificates, broken signing systems, and poorly secured account access as Figure 6 shows. A significant portion of this dataset deals with code signing and how attackers bypass its protections. They occur in both the development and design stage as well as at deployment with hijacked updates and compromised deployment servers, a sub-set discussed in detail in the next section.

Code signing is crucial to analyzing software supply chain attacks because, used correctly, it ensures the integrity of code and the identity of its author. Software supply chain

[56] Zetter, "How."

[57] Gross, "A Declaration."

[58] Peter Berkowitz, "The Lawyering of War," *Policy Review, February 1, 2020,* https://www.hoover.org/research/lawyering-war.

[59] Steven Musil, "Court Finds FBI Use of NSA Database Violated Americans' 4th Amendment Rights," CNET, October 8, 2019, https://www.cnet.com/news/court-finds-fbi-use-of-nsa-database-violated-americans-fourth-amendment-rights/; Charlie Savage and Jonathan Weisman, "N.S.A. Collection of Bulk Call Data Is Ruled Illegal," *New York Times, May 7, 2015,* https://www.nytimes.com/2015/05/08/us/nsa-phone-records-collection-ruled-illegal-by-appeals-court.html.

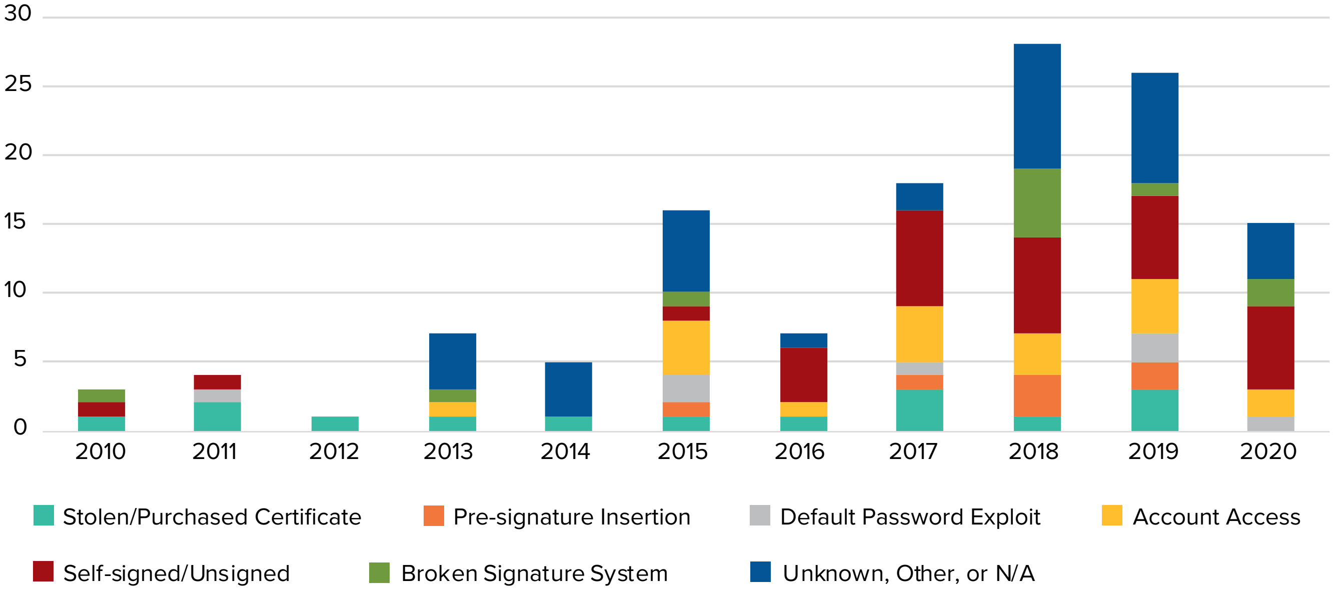


Figure 6. Changes in Attack Vector Over the Years

attacks rely on the attacker's ability to edit code, to pass it off as safe, and to abuse trust in the code's author in a software environment full of third-party dependencies (so many that vendors are usually unaware of the full extent of their dependencies). If code signing were unassailable, there would be far fewer software supply chain attacks as applications could assert code was unmodified and coming straight from the authentic source without concern.

But there are issues with code signing, so there are software supply chain attacks, and understanding the mechanism is crucial. Code signing is an application of public key cryptography and the trusted certificate system to ensure code integrity and source identity. When code is sent to an external source, it usually comes with a signature and a certificate. **The signature** is a cryptographic hash of the code itself that is encrypted with a private key. The consumer uses the public key associated with the code issuer to decrypt the hash and then hashes their copy of the code. If their hash matches the decrypted hash, they know their copy of the code is the same as the code that was hashed and encrypted by the author. The certificate is a combination of information that generally includes the name of the certificate authority (CA), the name of the software issuer, a creation and expiration date, and the public key associated with their private key. It provides the CA's assurance to the consumer that the issuer of the code has the private key that is cryptographically connected to the listed public key. **The software** issuer alone possesses the private key. The system's cryptography is similar to that of the certificate system behind TLS/SSL encryption (HTTPS connections), but distinct in its application. **(**Read more on the difference between code-signing certificate and SSL certificate here[60]).

There are several ways attackers can bypass these systems. The above description is a simplification. There is a chain of certificate authority dependencies: certificates can expire; there are a variety of algorithms used to sign code, some good and others bad; authors can fail to secure their private keys; vetting certificate requesters is challenging; and more. These complexities add vulnerabilities to the system. Attackers can modify code before it is signed by the issuer, meaning the code would be legitimately signed and still malicious. Attackers can compromise weak cryptography, which would allow them to forge a digital signature without needing to steal private keys. Attackers can even steal or buy private keys, allowing them to validly sign malware themselves. Sometimes organizations simply leak the private key by accident. Finally, some parts of computer architecture, like certain firmware drivers, were not designed with security in mind and do not even use code

[60] "Code Signing Certificate vs SSL Certificate: What's the Difference?" Comodo SSL Store, n.d., https://comodosslstore.com/resources/code-signing-certificate-vs-ssl-certificate-whats-the-difference/.

signing. All of these options are reflected in the dataset's attack vector variable.

For the purpose of this dataset, the difference between stealing private keys and certificates is negligible. The private key is the part of a certificate that is kept secret by the software issuer, and the part that is stolen when the phrase "stolen/purchased certificate" is used. The two phrases are interchangeable in most literature as well. A more technical breakdown can be found here,[61] but an attacker obtaining a legitimate certificate means that they got its associated private key.

Attacks that compromise the code signing system are extremely potent—they get a victim's system to run malicious code from an apparently legitimate source with significant authority to modify system files. As such, they have been on the rise recently and are a fundamental dimension of software supply chain attacks. (Read more about stolen certificate trends here[62] and here[63]).

Code-signing vulnerabilities are troubling because, if used successfully, they let attackers impersonate any trusted program and bypass some modern security tools. They also sow uncertainty about the authenticity of the very same patches and updates software developers use to fix security holes in their code. In January 2020, the United States' National Security Agency (NSA) broke with precedent and disclosed CVE-2020-0601 to Microsoft.[64] The bug is lodged in the Windows cryptographic libraries and allows attackers to forge the digital certificates that attest to an object's identity. Developers sign their code to authenticate it as their own. This gives users a way to identify known and authentic code from that which is unknown and possibly malicious. Browsing online to a bank? There is a digital certificate at the other end of the connection which validates the bank's website is what it purports to be. Running a software program from a company like Adobe or Microsoft on your computer? The operating system can compare that software's digital signature with the developers' to determine whether it is authentic or a malicious derivative. Code signing is an important process for establishing trust in the software supply chain.

The abuse of code signing is also not a recent phenomenon. In 2012, Adobe publicly warned about an attack on its systems that allowed attackers to create malicious files digitally signed with a valid Adobe certificate.[65] A malicious actor had compromised an internal Adobe build server, giving it access to internal code-signing infrastructure and the ability to create malware indistinguishable from legitimate Adobe software. One of the earliest examples of using code signing to disguise malware as legitimate software, the Adobe compromise begat a trend. Attacks using stolen or altered code signatures are littered throughout this paper's dataset.

In addition to stealing or forging certificates, attackers can also buy them from vendors online. These certificates could have been obtained through system access, but they can also come from mistakes (certificate authorities, or CAs, can be tricked into giving attackers legitimate certificates), insiders (employees with access to keys can sell them), or resellers (intermediate certificate issuers who require far less due diligence). The marketplaces are limited by the number of legitimate certificates issued, but they have been growing. The more expensive the certificate, the more trusted it is. The cost of a certificate is anywhere up to approximately $1,800 for an Extended Validation certificate issued by Symantec in 2016. CAs can revoke certificates that have been compromised and issue new ones, but the systems of reporting, intermediaries, and automatic revocation checks are slow to close security gaps. Moreover, without purchasing them off the black market and risking being defrauded, it is hard to know when certificates have actually been compromised until they are used in an attack. (Read more about certificate marketplaces here[66] and here[67]).

2.3 Breaking the Chain: Hijacked Updates

This section explores trends in where attackers targeted the software supply chain, focusing on the use of hijacked software updates. Across both attacks and vulnerabilities, supply chain service providers, third-party app stores, and hijacked updates are popular distribution vectors. Supply chain providers are instances where attackers insert their code in software used by a vendor who pass it on to

[61] Patrick Nohe, "Here's What Happens When Your Private Key Gets Compromised," hashedout, July 10, 2018, https://www.thesslstore.com/blog/heres-what-happens-when-your-private-key-gets-compromised/.

[62] Justine Brown, "Stolen Digital Certificates Are Hackers' Latest Weapon of Choice," CIODive, March 17, 2016, https://www.ciodive.com/news/stolen-digital-certificates-are-hackers-latest-weapon-of-choice/415804/.

[63] ENISA, "Valid Digital Certificates Code Signing Malware," European Union Agency for Cybersecurity, June 30, 2018, https://www.enisa.europa.eu/publications/info-notes/valid-digital-certificates-code-signing-malware.

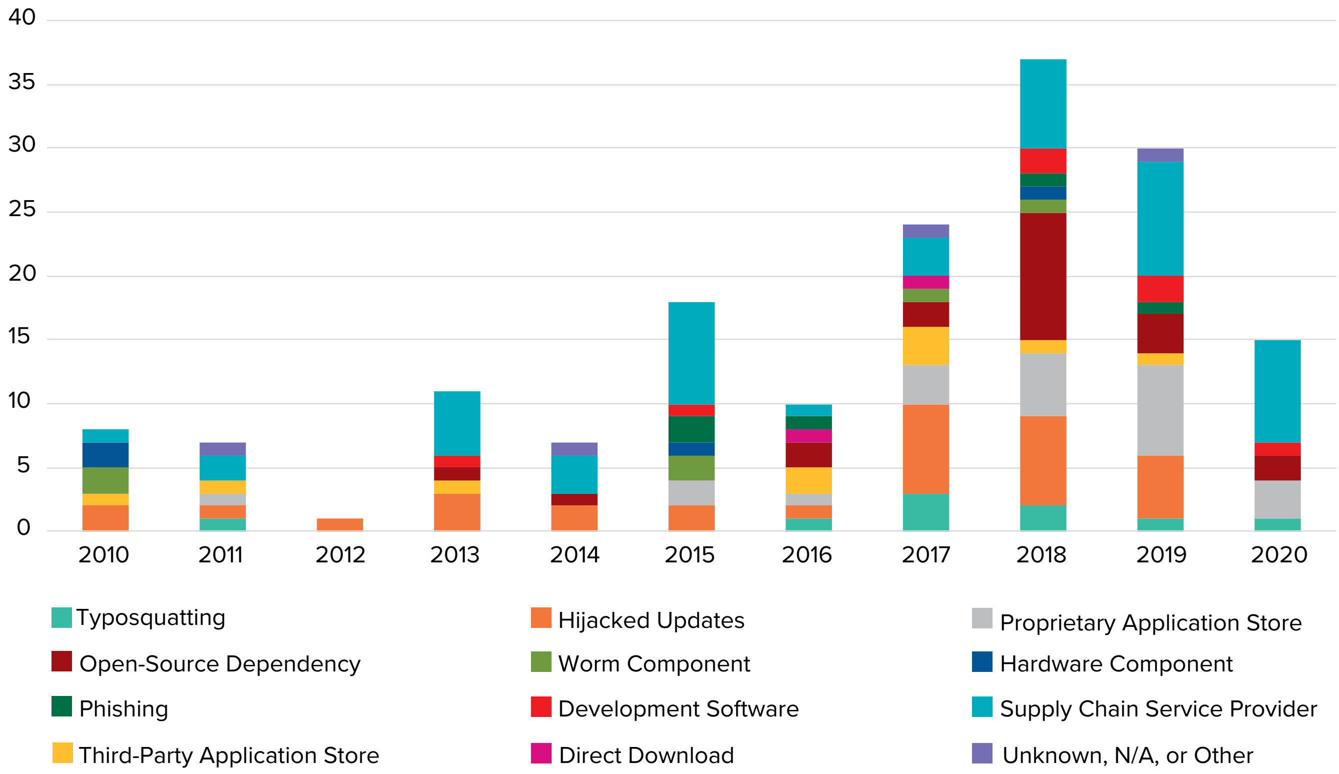
[64] CISA, "Alert (AA20-014A): Critical Vulnerabilities in Microsoft Windows Operating Systems," Cyber and Infrastructure Security Agency, January 14, 2020, https://us-cert.cisa.gov/ncas/alerts/aa20-014a.

[65] Naraine, "Adobe."

[66] John Leyden, "Stuxnet-Style Code Signing of Malware Becomes Darknet Cottage Industry," *Register, November 4, 2059,* https://www.theregister.com/2015/11/04/code_signing_malware/.

[67] Limor Kessem, "Certificates-as-a-Service? Code Signing Certs Become Popular Cybercrime Commodity," Security Intelligence, September 9, 2015, https://securityintelligence.com/certificates-as-a-service-code-signing-certs-become-popular-cybercrime-commodity/.

Figure 7. Changes in Distribution Vector Over the Years



customers, for example, HVAC climate control services, firmware providers, or maintenance and service firms. These providers enable malicious code to spread by connecting malware to targets via intermediary services: from 5G providers to creators of intermediary software running PDF viewers to the authors of mobile apps, ICS systems, and car media player programs. Figure 7 shows the change in distribution vector for attacks over time.

Software updates are a major trust mechanism in the software supply chain and they are a popular vector to distribute attacks relying on compromised or stolen code-signing tools. Hijacking update processes is a key component of the more complex and harmful software supply chain attacks. The majority of hijacked updates in this report's dataset were used to spread malicious code to third-party applications as new versions of existing software and open-source dependencies spread malware inserted into open-source libraries. One example is GOM Player, a free media application. On January 2, 2014, suspicious activity was detected in the reactor control room of the Monju fast breeder reactor facility in Tsuruga, Japan.[68] A routine software update for the free media player GOM Player, a popular substitute for Windows Media Player, especially in parts of Asia, had been compromised to include malware, which gave the attacker access and the capability to exfiltrate information. In the case of the Monju reactor facility, this information was more than 40,000 internal emails.

The Monju attack demonstrated the pernicious ease of a compromised update. Attackers were able to capitalize on the trust users placed in updates. Compromise required little more than the press of a button on a seemingly legitimate update to spread the attack. Despite likely not being an intentional target, the Monju reactor facility fell victim to an attack on a widely popular but simple media player—underlining how the tangled nature of software supply chains can easily bring innocuous code to high-consequence targets. Figure 8 illustrates the frequency of different attack and distribution vector combinations in our dataset.

Hijacking updates generally requires accessing a certificate or developer account, versus app store and open-source attacks which can rely on unsigned attacker-made software. CCleaner is a great example. In September

[68] Mark Graham, "Context Threat Intelligence—The Monju Incident," *Context Threat Intelligence, February 19, 2014,* https://www.contextis.com/en/blog/context-threat-intelligence-the-monju-incident.

Figure 8. Map of Attack Vector by Distribution Vector

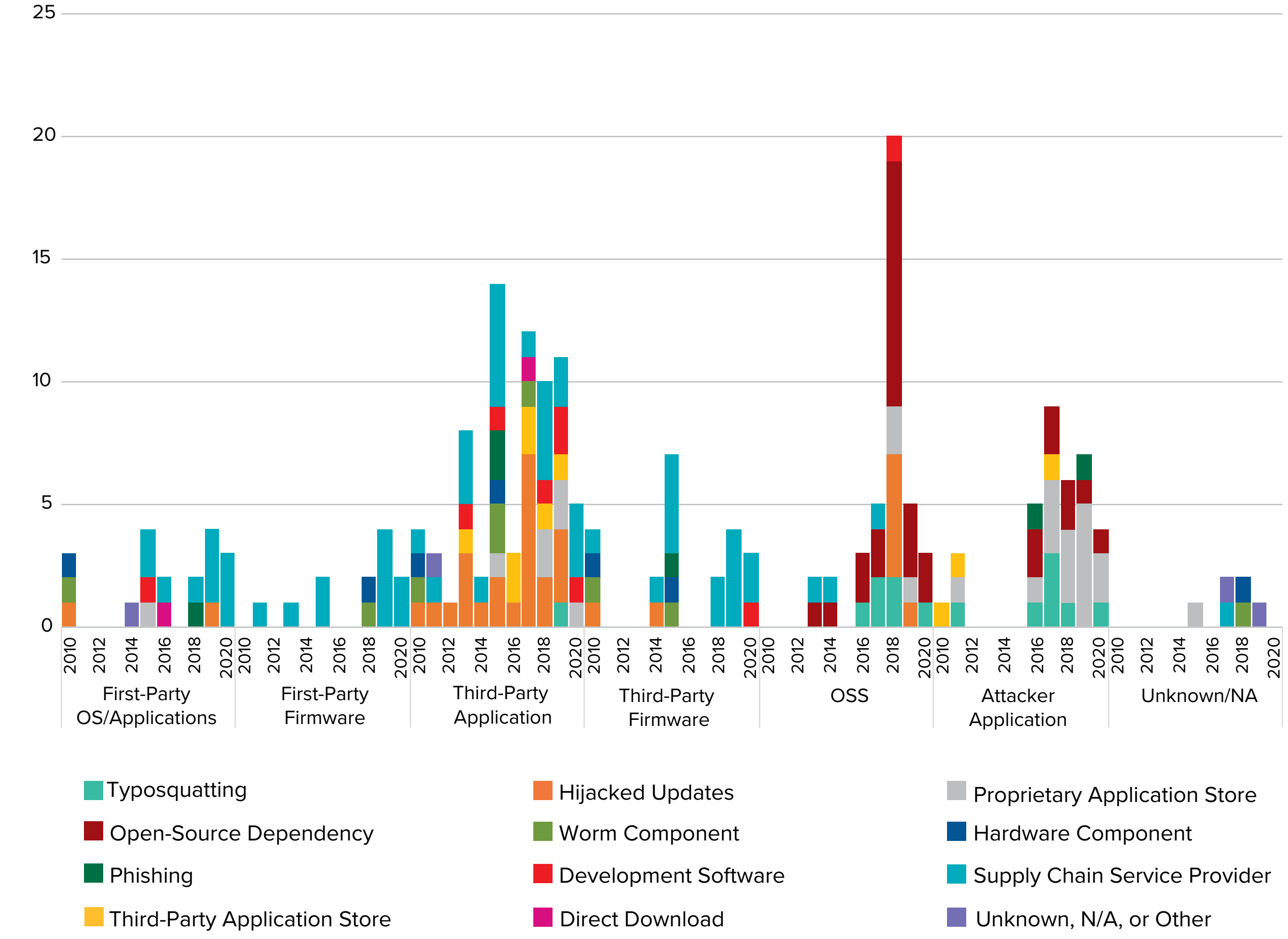
Attack Vectory by Distribution Vector	Stolen/ Purchased Certificate	Pre-signature Insertion	Default Password Exploit	Account Access	Self-signed/ Unsigned	Broken Signature System	Unknown, Other, or N/A
Typosquatting	0	0	0	1	**8**	0	0
Hijacked Updates	**13**	1	0	**9**	3	2	7
Proprietary Application Store	0	3	0	2	**17**	1	2
Third-Party Application Store	1	1	0	0	6	2	0
Open-Source Dependency	0	1	0	**6**	**9**	0	6
Worm Component	1	0	0	2	0	1	2
Hardware Component	1	0	0	1	0	1	1
Direct Download	1	0	0	0	0	0	1
Phishing	0	0	0	2	2	0	2
Development Software	1	5	0	0	2	0	2
Supply Chain Service Provider	5	1	7	**7**	2	**10**	**23**
Unknown, N/A, or Other	1	0	0	1	0	0	3

2018, Cisco Talos researchers disclosed that the computer cleanup tool had been compromised for several months.[69] Attackers had gained access to a Piriform developer account and used it to obtain a valid certificate to update the CCleaner installer to include malicious code that sent system information back to a command and control (C&C) server. The attack initially infected more than two million computers and downloaded a second, stealthier payload onto selected systems at nearly two dozen technology companies.

Across the data we collected, hijacked updates were most commonly used to target third-party applications like CCleaner. While attackers targeting CCleaner penetrated its developers' networks and stole their signing certificates, there are other examples of brute force abuse. The Flame

[69] Newman, "Inside."

Figure 9. Changes in Targeted Code by Distribution Vector Over the Years



malware, discovered in 2012, leveraged an MD5 hash collision[70] to forge semi-valid code signatures.[71] Code-signing certificates are also available on underground markets. The most trusted, secure certificates go for as little as $1,600[72]—a drop in the bucket for nation-state attackers, and not prohibitively expensive for criminal enterprises. Figure 9 shows this distribution vector by targeted codebase relationship changing over time across this report's dataset. The relationship in third-party code remains consistent but this shows a more episodic pattern with first-party OS/applications and open-source software.

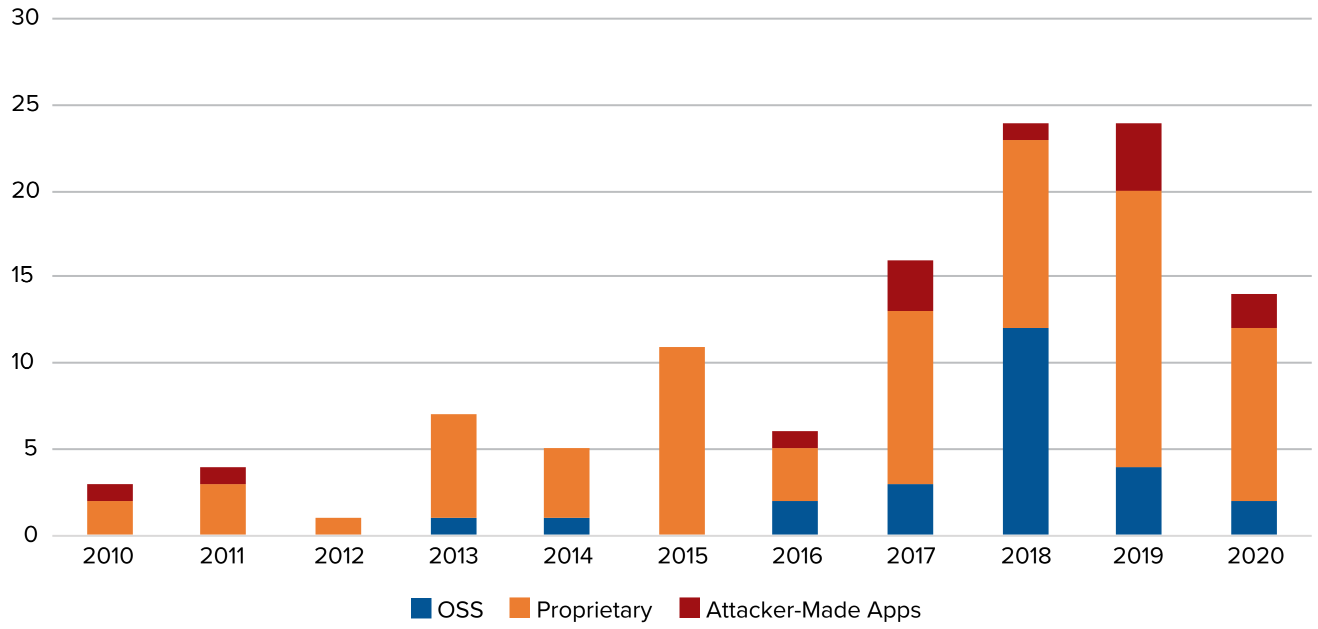
Hijacked updates rely on compromising build servers and code distribution tools. Rarely do attacks involve brute force assaults against cryptographic mechanisms. The implication is that many of the same organizational cybersecurity practices used to protect enterprise networks and ensure limited access to sensitive systems can be applied, if only more rigorously, here to address malicious updates. The update channel is a crucial one for defenders to distribute patches to users. Failure to protect this linkage could have dangerous ripple effects if users delay applying, or even begin to mistrust, these updates.

[70] Swiat, "Flame."

[71] Sotirov, "Analyzing."

[72] Dan Goodin, "One-Stop Counterfeit Certificate Shops for All Your Malware-Signing Needs," Ars Technica, February 22, 2018, https://arstechnica.com/information-technology/2018/02/counterfeit-certificates-sold-online-make-digitally-signed-malware-a-snap/.

Figure 10. Changes in Affected Code Types Over the Years



2.4 Poisoning the Well: Open-Source Software

As software continues to spread at an unprecedented pace, developers are under pressure to create new products and services ever faster and at lower cost. Open-source software is a crucial layer in the software ecosystem that needs more effective protection and is under-addressed especially as many vulnerabilities are hidden under layer after layer of dependencies. Looking across the software supply chain, there are a variety of types of codebase interleaving in a complex web of dependencies. Even larger-scale proprietary projects like the Windows operating system integrate large amounts of open-source code. How these projects are built and managed can shed light on the feasibility of widely adopting best practices for things like code integrity and long-term updates.

There are two significant, and distinct, cultures of software development: open-source and proprietary. Both shaped by the principles of their communities and the legal status of their products. Proprietary software is code owned by a single individual or organization.[73] Some of the most well-known examples of proprietary code are Microsoft Windows, Adobe Flash Player, Skype, and Apple's iOS. Apple's iOS is developed in house, a product of the Cupertino giant's design, market research, and myriad development teams. Thus, the vast majority of software produced for iOS is controlled, updated, licensed, and sold by Apple without outside development.

In contrast, an open-source community is defined as "an interacting, self-governing group involved in creating innovation with members contributing toward a shared goal of developing free/libre innovation."[74] Open-source developers voluntarily collaborate to develop software that is valuable to them or their organization. Open source has become the bedrock of technological innovations like cloud computing, software-as-a-service, next generation databases, mobile devices, and a consumer-focused Internet.[75] Figure 10 aggregates this evolving attack surface by looking at just attacks and vulnerabilities targeting proprietary versus OSS codebases.

Most open-source projects do *not* follow strict organizational structures, instead relying on self-organization and a collaborative approach to drive software innovation and development. Open-source projects "give away" part of their

[73] "Tell Me More About Proprietary Software Code," Metis Partners, accessed June 15, 2020, http://metispartners.com/ip-basics/informal-ip/proprietary-software-code/.

[74] Michael Ayukawa, Mohammed Al-Sanabani, and Adefemi Debo-Omidokun, "How Firms Relate to Open Source Communities," *Technology Innovation Management Review, January 2011,* https://timreview.ca/article/410.

[75] Peter Levine, "Why There Will Never Be Another RedHat: The Economics Of Open Source," Tech Crunch, February 13, 2014, https://techcrunch.com/2014/02/13/please-dont-tell-me-you-want-to-be-the-next-red-hat/.

The Left-Pad Affair

Despite the destructive potential of software supply chain attacks, little has been done to impose a cost on even the most basic; some attacks happen purely by accident. An npm publisher, Azer, removed 273 packages from the online JavaScript repository, including left-pad, a simple but commonly relied on program that pads a string with zeros on its left. So many programs relied on Azer's left-pad, or programs that relied on left-pad, or programs that relied on programs relying on left-pad (ad nauseam) that hundreds of failures per minute occurred before npm republished the package two hours later, leading some users to claim that Azer "broke the Internet."

intellectual property in order to create and benefit from a larger marketplace of ideas.[76] Open source has given rise to a complex social web, with some supportive of limited and general copyrights even for code which is never sold, and others objecting to anything less than free software. Richard Stallman, founder of the GNU project and co-founder of the League for Programming Freedom, is famously quoted as saying "free software is a political movement; open source is a development model."[77] Stallman believes that, like speech, people should be free to code whatever and whenever they want.

There are two major types of OSS. Project or community-based OSS is the most common example: a distributed community of developers who continuously update and improve a codebase.

Ruby is a classic example of open-source development. It is a community-based open-source codebase created in 1995 by Yukihiro "Matz" Matsumoto with a focus on "simplicity and productivity."[78] Twitter, Hulu, Shopify, and Groupon are just a few well-known sites built with Ruby. Individuals can manage their own packages and dependencies on a day-to-day basis to ensure their quality. Using a collection of package, version, and gem managers, as well as the web framework Ruby on Rails, the Ruby codebase is diverse and constantly growing. Attacks on Ruby feature in four different incidents in the dataset including a March 2019 attack on the "strong_password" gem which inserted a backdoor into code used to evaluate the strength of passwords on websites. The gem was downloaded more than 500 times before a single developer auditing the code noticed the change.

The second, less intuitive type of open-source project is commercial open-source software (COSS). The major difference between the two is that COSS has an owner with full copyright, patents, and trademarks despite its development by a broader community. Up until its purchase by IBM in 2019, Red Hat was the largest COSS entity in the world. The company runs and operates an eponymous distribution (version) of the Linux operating system. Linux has existed since 1991 and is found[79] in everything from cars and home appliances to supercomputers. Red Hat maintains profitability by giving away its OSS, but charging customers for support, maintenance, and installation.[80]

While proprietary code is owned by a single entity code and generally sold for profit, it can include open-source elements either directly or as part of a network of software around the product. MacOS is famously based in large part on the FreeBSD (Berkeley Software Distribution) version of Unix, and Windows 10 now includes a Microsoft-developed version of the full Linux kernel.[81]

The open-source development model is extremely flexible and can achieve all sorts of software functionality through a multitude of different development communities. Apache software is developed and maintained by the Apache Software Foundation and is the most widely used free web server in the world, running 67 percent of all web servers.[82] OpenSSL is a software toolkit that implements the Transport Layer Security (TLS) and Secure Sockets Layer (SSL) protocols The project is maintained by designated management and technical committees, and

[76] J.V. Joshua, D.O. Alao, S.O. Okolie, and O. Awodele, "Software Ecosystem: Features, Benefits and Challenges," *International Journal of Advanced Computer Science and Applications (2013) Vol. 4, No. 8: 242-247, accessed July 15, 2020,* https://thesai.org/Downloads/Volume4No8/Paper_33-Software_Ecosystem_Features,_Benefits_and_Challenges.pdf.

[77] Richard Stallman, "Why Software Should Be Free," GNU Operating System, accessed July 15, 2020, https://www.gnu.org/philosophy/shouldbefree.html.

[78] Kimberley Cook, "Python vs. Ruby: Which Is Better for Every Programmer and Why?" House of Bots, November 6, 2018, https://www.houseofbots.com/news-detail/3957-1-python-vs-ruby-which-is-better-for-every-programmer-and-why.

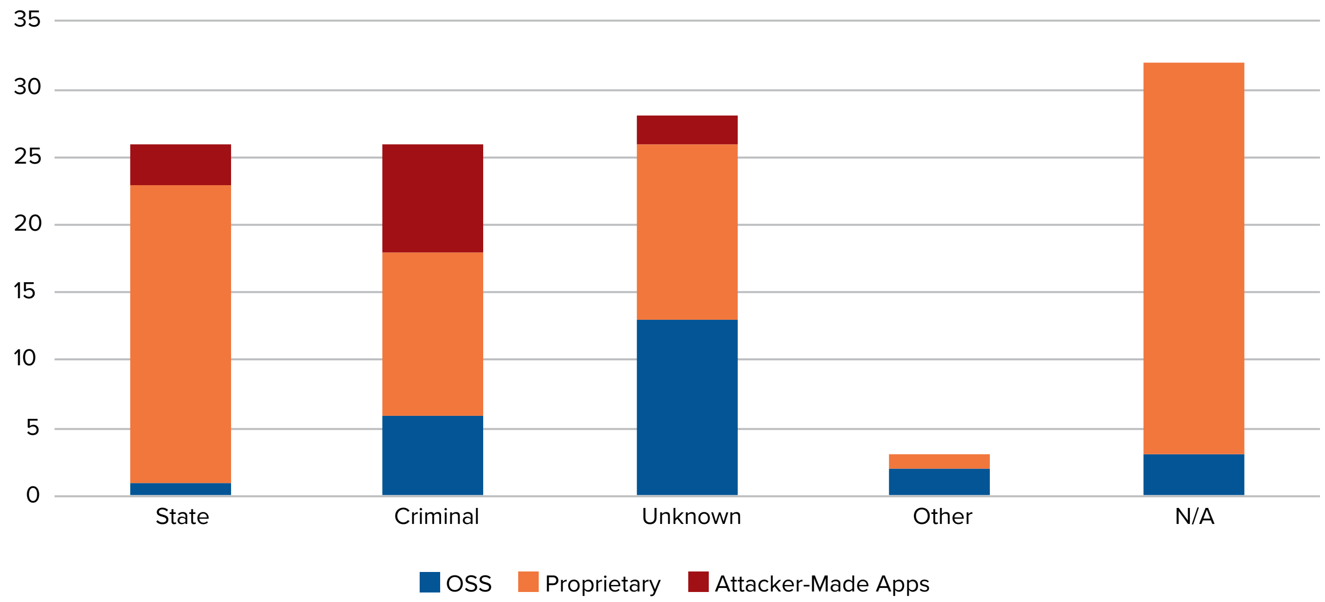
[79] Jenni McKinnon, "A Citizen's Guide to Open Source Communities," Pagely, April 18, 2019, https://pagely.com/blog/citizen-guide-open-source-community/.

[80] Levine, "Why."

[81] Mary Jo Foley, "Windows 10 is Getting a Microsoft-Built Linux Kernel," ZDNet, May 7, 2019, https://www.zdnet.com/article/windows-10-is-getting-a-microsoft-built-linux-kernel/.

[82] McKinnon, "A Citizen's."

Figure 11. Targeted Code Type by Attacker Type



all development is meant to follow a set of established bylaws. OpenSSL does emphasize security bug reporting.[83] Large-scale vulnerabilities in OpenSSL have been disclosed in the past—most notably, Heartbleed.[84] Although Heartbleed was a serious vulnerability that gave malicious individuals access to hundreds of computers, it did have one positive side effect—OpenSSL received more than $300,000 in funding from major tech companies to hire new full-time developers.[85] After years of operating while severely underfunded and understaffed, this kind of funding was an important first step for the open-source project. Figure 11 illustrates the distribution of targeted code types, including open-source software, by different types of attackers.[86]

Attacks on OSS have grown more frequent in recent years. In February 2020, two accounts uploaded more than 700 packages to the Ruby repository and used typosquatting to achieve more than 100,000 downloads of their malware, which redirected Bitcoin payments to attacker-controlled wallets. 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.

2.5 Downloading Trouble: App Hubs/Stores under Attack

App hubs/stores were a popular means of disseminating software supply chain attacks in our analysis. The stores are a common feature of the software ecosystem and how many users interact with the software supply chain on a regular basis. They serve as a marketplace for products and updates from third-party developers. Some apps fulfill specific functions, like advanced graphing calculators or audio editing suites. Others integrate different services like the Google Drive and Gmail apps for iOS. A few enable features relying on access to multiple devices, like two-factor authentication. Still more provide a window to massively popular platforms like Facebook, Instagram, and Snapchat, which are sometimes only accessible from mobile devices. As a consolidated venue, app stores simplify the user's search for software that maximizes the value of their devices. However, as a high-traffic download center that connects end users to third-party developers, app

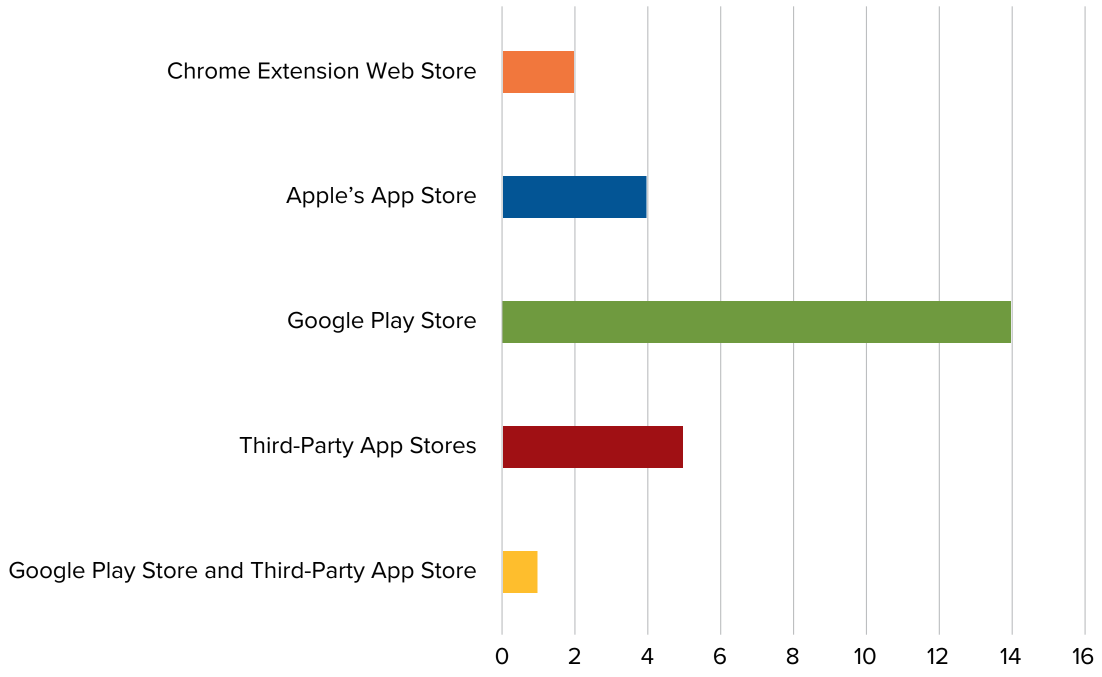
[83] Community, Open SSL, accessed June 15, 2020, https://www.openssl.org/community/.

[84] Timothy B. Lee, "The Heartbleed Bug, Explained," *Vox, May 14, 2015,* https://www.vox.com/2014/6/19/18076318/heartbleed.

[85] Jack Wallen, "From Underfunded to Funded Within a Heartbleed," Tech Republic, April 25, 2014, https://www.techrepublic.com/article/from-underfunded-to-funded-within-a-heartbleed/.

[86] Jose Pagliery, "Your Internet Security Relies on a Few Volunteers," CNN, April 18, 2014, https://money.cnn.com/2014/04/18/technology/security/heartbleed-volunteers/index.html.

Figure 12. Incidents by App Store Type



stores have become a popular way to attack the software supply chain. Figure 12 shows the number of incidents we observed for different app stores.

Improving the security of software available through major app hubs like Google Play Store and Apple's App Store must be a priority for developers and the software industry. The vulnerability of these hubs has long been recognized.[87] It is an increasing source of risk to the national security enterprise as relatively innocuous services like Strava can be used to compromise the location, health, and welfare of military personnel.[88] The app store model continues to weather a storm of attacks due to the difficulty of vetting the third-party products at its heart. Malware obfuscation techniques have continued to evolve, and the volume of apps that need screening has also increased.

App store attacks generally unfold in one of three ways. Attackers can build their own apps, designed to appear legitimate, perhaps providing wallpapers, tutorial videos, or games. Hidden in those applications, which might function as advertised, is malicious software. Sometimes when attackers create their own apps, they try to impersonate legitimate ones either through typosquatting or by posing as updates (the difference being that the latter will not provide functionality to the user). One example is the ExpensiveWall attack, named after one of several apps that attackers uploaded to the Google Play Store, Lovely Wallpaper, which provided background images for mobile users.[89] The malicious software in the apps evaded Google Play's screening system using encryption and other obfuscation techniques. The malware, once downloaded, charged user accounts and sent fraudulent premium SMS messages, driving revenue to the attackers. It could also be easily modified to extract sensitive data and even microphone recordings from victims' machines. The malware was eventually found in more than 50 apps cumulatively downloaded between 1 and 4.2 million times before they were removed from the store (installed apps remained on devices until users opted to remove them).

[87] Avi Bashan, "Mobile Security: Why App Stores Don't Keep Users Safe," Dark Reading, March 24, 2016, https://www.darkreading.com/vulnerabilities---threats/mobile-security-why-app-stores-dont-keep-users-safe/a/d-id/1324829; Catalin Cimpanu. "Open-Source Spyware Makes It on the Google Play Store," ZDNet, August 22, 2019, https://www.zdnet.com/article/open-source-spyware-makes-it-on-the-google-play-store/.

[88] Jeremy Hsu, "The Strava Heat Map and the End of Secrets," *WIRED, January 29, 2018,* https://www.wired.com/story/strava-heat-map-military-bases-fitness-trackers-privacy/; Foeke Postma, "Military and Intelligence Personnel Can Be Tracked with the Untappd Beer App," Bell?ngcat, May 18, 2020, https://www.bellingcat.com/news/2020/05/18/military-and-intelligence-personnel-can-be-tracked-with-the-untappd-beer-app/; "Threat Research," VMWare Carbon Black, accessed June 15, 2020, https://www.carbonblack.com/threat-research/; Niraj Chokshi, "Hack of Quest Diagnostics App Exposes Data of 34,000 Patients," *New York Times*, December 12, 2016, https://www.nytimes.com/2016/12/12/us/hack-of-quest-diagnostics-app-exposes-data-of-34000-patients.html?_r=0.

[89] Check Point, "ExpensiveWall."

Attackers can also repackage apps, meaning they take a legitimate app, add their own malicious code, and then bundle it as a complete item for download, usually on third-party sites. In one case, attackers repackaged Android versions of Pokémon Go to include the DroidJack malware, which could access almost all of an infected device's data and send it to an attacker C&C server by simply asking for extra app permissions.[90] The attackers placed the malware-infested app on third-party stores to take advantage of the game's staggered release schedule, preying on users looking for early access to the game.

There is also publicly available evidence that groups compromise the software used to build apps, also known as software development kits (SDKs), allowing them to inject malware into legitimate apps as they are created. Whatever the model, malicious actors make an effort to obfuscate their malware to evade detection by app store curators. 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.[91] Xcode is a development environment used exclusively to create iOS and OS X apps. Typically, the tool is downloaded directly from the App Store, but it is available elsewhere on the web outside Apple's review. A version of Xcode found on Baidu Yunpan, a Chinese file-sharing service, came embedded with malicious logic that modified apps created in the development environment. Each app created with XcodeGhost relayed information about a customer's device once the app had been downloaded, such as time, location, name of device, and network type. It also allowed attackers to phish credentials, open URLs, and access a device's clipboard, which can often contain password and payment information. This version of Xcode was later dubbed XcodeGhost.

Multiple apps created with XcodeGhost were accepted into Apple's App Store, including malicious versions of WeChat, WinZip, and China Unicom Mobile Office, eventually impacting more than 500 users. XcodeGhost aptly demonstrates the significance of app stores as a distribution method. While Apple's App Store is generally more rigorously policed, we recorded four successful attacks targeting it compared with more than a dozen against Google Play Store and other third-party distributors. These venues are attractive targets as the sheer scale of the attack's blast radius demonstrates.

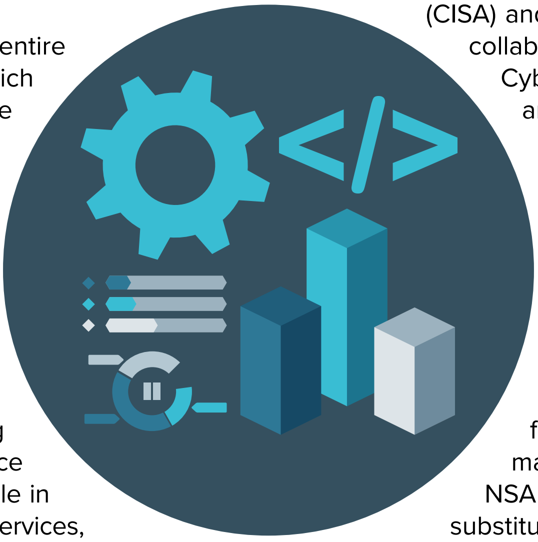
[90] Chris Brook. "Malicious Pokémon Go App Installs Backdoor on Android Devices," Threat Post, July 11, 2016, https://threatpost.com/malicious-pokemon-go-app-installs-backdoor-on-android-devices/119174/.

[91] 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/.

3. Recommendations

Software supply chain attacks remain popular, impactful, and are being used to great effect by states. The sustained growth of software supply chain attacks is caused at a technical level by continued failure to secure code integrity. Attackers continue to find ways to access accounts and bypass code signing, app stores struggle to verify the innocuity of all their software, developers embed insecure design choices at the lowest level of computing, and vendors have difficulty fully grasping the scope of their software dependencies and reliance on supply chain service providers. These are complex technical challenges with neither easy nor immediate solutions, and they further complicate the lapse in policy progress to secure a supply chain that has grown critical to industry and national security.

The most disconcerting trend of this entire dataset is the consistency with which these attacks occur against sensitive portions of the supply chain—this is not a new problem. A 2010 report from Carnegie Mellon University's Software Engineering Institute profiled the DoD's concern that "security vulnerabilities could be inserted into software."[92]



Progress to improve the security of these supply chains has been halting for a multitude of reasons. Open-source projects continue to play a central role in enabling new software products and services, and fill critical gaps in the security architecture of the Internet. Despite this, efforts to better resource and secure these projects, like the Linux Foundation's Core Infrastructure Initiative, remain too few and under-resourced relative to the problem. Standards and existing security tools are too often applied with a "check-once-at-one-point-in-time" mindset. Scanning for vulnerabilities and working to remediate them is a constant process. Auditing the trust of an update channel, verifying the integrity of new code in a production environment, and assessing risks from third parties and open-source packages must be ongoing activities.

Policy efforts which proliferate practices, protocols, standards, and codes of conduct which can't be automated or which are not straightforward to implement at compile or commit time are insufficient to the problem. Software developers thrive on tools and dynamic means to implement controls on their code. Some public sector agencies have begun to better acknowledge this. The NSA recently open sourced a well-featured reverse engineering tool called Ghidra alongside a host of small programs to ingest and analyze large security datasets. This is far from the norm, however, and more can be done both to tie security standards efforts to automation and tooling as well as push for more DoD, Department of Homeland Security (DHS), and NSA tools to be open sourced and made publicly useful.

Many of these recommendations focus on the role of DHS' Cybersecurity and Infrastructure Security Agency (CISA) and could be improved through CISA's collaboration with the NSA's still new Cybersecurity Directorate. While there are flashes of sibling rivalry between the two, together they would leverage a significantly deeper pool of technical expertise and security acumen than alone. Failure to collaborate would impact not only efforts to improve open-source security but also the proposed improvements to baseline software supply chain security and efforts to counter systemic threats. In many cases we advocate for a CISA or NSA role in particular efforts but there is substitutability.

The United States and its allies cannot afford drive-by reforms. Sustained improvement is necessary. The power of the software supply chain is how it enables rapid and often low-cost change in the functionality of complex systems. This same rapidity can be the undoing of users and organizations. Policy makers should focus on supporting industry efforts to improve the supply chain security baseline and collaborate to reduce the impact of state attacks. Risk is the name of the game—not bought completely away or mitigated through whiz bang technology but managed deliberately, thoughtfully, and consistently. In this the report owes an intellectual debt to numerous prior efforts but particularly three reports, the New York Cyber Task Force's

[92] Robert J. Ellison, John B. Goodenough, Charles B. Weinstock, and Carol Woody, "Evaluating and Mitigating Software Supply Chain Security Risks," Carnegie Mellon University's Software Engineering Institute, May 2010, https://resources.sei.cmu.edu/library/asset-view.cfm?assetid=9337.

Building a Defensible Cyberspace[93], MITRE's *Deliver Uncompromised*[94], and the Carnegie Endowment Cyber Policy Initiative's *ICT Supply Chain Integrity: Principles for Governmental and Corporate Policies*[95].

Software will continue to "eat" the world and these recommendations are meant to support that—enabling more effective risk management and improved security practices broadly to raise the baseline cost of software supply chain attacks and reduce the impact of the most consequential:

3.1 Improve the Baseline

Software supply chain attacks and disclosures that exploit vulnerabilities across the lifecycle of software in this dataset are too cheap relative to the harm they can impose. For example, while software updates are *the* critical channel to bring updates and patches to software users, they have been hijacked consistently over the past decade. The problem is generally not developers brewing their own cryptographic protections or poorly implemented accepted protocols but failing to rigorously implement basic protections for their networks, systems, and build servers.

This cluster of recommendations is intended to reduce the complexity and burden of implementing secure software supply chain practices across organizations and software projects of all sizes. The central pillar is a new "Life Cycle Security Overlay" for Special Publication (SP) 800-53 developed by the National Institute of Standards and Technology (NIST), which integrates other identified industry best practices. SP 800-53 is one of the most complete bodies of security controls and has global visibility through the Cybersecurity Framework. SP 800-53 is not perfect; it locks in an impact-centric model of auditing and assurance and often trades compliance for risk management, but it is a tool in hand and far more useful today than an as-yet undeveloped specification many years out.

Raising the cost of software supply chain attacks should center on providing the whole of industry, from SAP and Microsoft down to a three-person LiDAR startup, easy-to-use tools and well-defined reference implementations for major cloud and IT vendors that make rigorous security as low-effort and cheap as possible. Efforts to build an assessable maturity model, identify reference implementations for this Overlay, and build an open-source tooling to support these implementations all build from the Overlay itself. These would provide resources to developers as well as a framework to measure software supply chain security performance in federal contracting, opening the possibility that such measures could trickle down into the private sector and be enforced within segments of the technology marketplace. These steps also support implementation of recommendations made by others on improving operational coordination and developing security metrics necessary to "build solutions that scale at least cost for greater security."[96] Finally, this section calls for an organization to maintain the value of these tools over time, collecting, curating, and caring for the most useful long into the future.

FIRST

1. **Life Cycle Security Overlay** *[NIST and Industry]***:** Develop a software supply chain security Overlay to NIST SP 800-53, wrapping in controls from existing families, the new supply chain family in 800-53 rev5, and best practices collected in the Secure Software Development Framework (SSDF) and related industry and open-source publications like the BSA Framework for Secure Software.[97] This recommendation builds on the strong network and expertise of NIST and follows on previous recommendations to anchor technical security obligations in standard-setting organizations.[98]

    a. **Sector-Specific Agencies Implement Overlay** *[NIST and SSA Working Groups]***:** The NIST Overlay team should support appropriate sector-specific agencies to set up implementation working groups with industry partners focused on using this Overlay in their own development and contracting with third parties. NIST should feed requests

[93] New York Cyber Task Force. *Building a Defensible Cyberspace, Columbia University, September 2017,* https://sipa.columbia.edu/sites/default/files/3668_SIPA%20Defensible%20Cyberspace-WEB.PDF.

[94] Christopher Nissen, John Gronager, Robert Metzger, Harvey Rishikof, *Deliver Uncompromised*, MITRE Corporation, August 2018, https://www.mitre.org/publications/technical-papers/deliver-uncompromised-a-strategy-for-supply-chain-security?utm_source=P&utm_medium=P&utm_campaign=ieee

[95] Ariel (Eli) Levite, *ICT Supply Chain Integrity: Principles for Governmental and Corporate Policies, Carnegie Endowment for International Peace, October 4, 2019,* https://carnegieendowment.org/2019/10/04/ict-supply-chain-integrity-principles-for-governmental-and-corporate-policies-pub-79974.

[96] New York Cyber Task Force. *Building a Defensible Cyberspace*

[97] The Council on Foreign Relations has similarly highlighted the need for affected vendors to receive "specific, targeted threats and technical indicators," and for US policy makers to "facilitate more actionable cyber-threat information sharing, including informing vendors when intelligence agencies find vulnerabilities in supply chains or products," in order for vendors to appropriately defend their supply chains (*Improving Supply-Chain Policy for U.S. Government Procurement of Technology*).

[98] Ariel (Eli) Levite, *ICT Supply Chain Integrity: Principles for Governmental and Corporate Policies*

for more specific controls or guidance into an 18-month revision cycle, producing additional guidance or changes to the Overlay as needed, for example, for industrial control systems in the energy sector.[99]

2. **Bring the Overlay to the Cloud** *[Industry, Especially Cloud Service Providers]***:** Many software developers rely in whole or in part on cloud vendors to host, distribute, and maintain their codebases. Industry can assert moral leadership on software supply chain security issues and realize practical financial advantages by offering public reference implementations of the Overlay in their services and lower the complexity of secure life cycle practices for customers. Major cloud providers, namely Amazon, SAP, Microsoft, Google, Dell, and IBM, should build on existing industry organizations and collaboration to lead joint development of these reference implementations and make them freely available on government and industry partner websites.

   a. **Integrate and Grow the SBOM** *[NIST and the National Telecommunications and Information Administration]***:** NIST and NTIA should integrate the draft Software Bill of Materials (SBOM) standard developed by the NTIA multi-stakeholder process into this reference implementation material. NTIA should continue to evangelize on the role and utility of software transparency, leading a standing multi-stakeholder working group on SBOM.

3. **Release the Tools!** *[DoD and Industry]***:** Companies and open-source projects, including the cloud providers above, should commit to release tooling to help implement the Overlay. These industry efforts should be joined in parallel by the office of the DoD Chief Information Officer, the Defense Digital Service, and the Defense Information Security Agency as well as any other significant software development efforts within the federal government. An encouraging sign of the DoD's commitment to this policy would be a new instruction from the secretary of defense modifying and updating DODI 8500.01 accordingly.

NEXT

4. **Harmonize the Overlay** *[NIST, DHS CISA, and ENISA]***:** NIST, DHS CISA, and ENISA should work with industry partners and the international standards community to map this Overlay to appropriate ISO controls and the EU Cybersecurity Certification Framework to harmonize standards across the transatlantic community.

5. **Recognize Software is Part of 5G** *[State Department and NSC]***:** The State Department and the NSC's efforts to shape a like-minded coalition for the United States on 5G security issues has produced variable results but is driving strong industry, and some international, pressure for a more open telecommunications technology marketplace. Much of that push is a result of industry trends towards virtualizing complex hardware functions in software. Starting with the EU, the State Department and NSC should work with civil society and industry to make software development and supply chain security principles a core part of the 5G coalition-building process.

OVER THE HORIZON

6. **Measure Overlay Maturity** *[NSA and Industry]***:** NSA's Cybersecurity Directorate should work with an appropriate industry consortium to develop a software supply chain security maturity model based on this control Overlay and make it publicly available.

   a. **Tie Overlay to CMMC** *[DoD]***:** The DoD should integrate this supply chain maturity model as part of its Cybersecurity Maturity Model Certification (CMMC) program and establish a level of performance required

[99] The Commission on Enhancing National Cybersecurity, in its *Report on Securing and Growing the Digital Economy*, similarly urged NIST to conduct research on supply chain risk focused on organizational interdependencies, recommending that it "identify methods that assess the nature and extent of organizational interdependencies, quantify the risks of such interdependencies, and support private-sector measurement against standards of performance." Various organizations, including the New York Cyber Task Force and Northrop Grumman, have highlighted the importance of ensuring that private sector entities implement NIST standards and voluntary practices, for instance, by making them more accessible for all stakeholders. (*Raising the Bar on Cybersecurity and Acquisition*).

for prime contractors. The DoD should further implement these performance measures as new contracting requirements for information technology procurement.

b. **Assess Overlay Performance in IT Contracting** *[General Services Administration]***:** GSA should establish similar performance measures against this maturity model and implement them as part of evaluating new federal information technology contracts.

3.2 Better Protect Open Source

One of the popular distribution vectors for software supply chain attacks in this report's dataset was open-source packages and libraries. These are rarely the most consequential attacks, but they are often exploited through trivial effort, pointing to a concerning trend given the wide dependence on open-source code in commercial and national security applications. Continuing efforts by the White House to incorporate open-source software as a means of sharing code across different agencies and with the technical public raise the stakes of securing open-source software development.[100]

This cluster of recommendations aims to support more effective and consistent security practices across open-source projects and in the governance of repositories and code registries. Policy makers should *not* endeavor to "fix" the open-source community. There is no one open-source community and effective change comes from resources, tools, education, and time—not trying to upend cultures.

These recommendations provide additional resources for security in the open-source community, including grant funding, public sector support and policy evangelism for best practices, and industry incentives to build in tools for supply chain security to hubs and repositories.[101] Executed together, these changes should improve the health of open-source software, use federal funding to support private sector leverage over attackers,[102] and help raise the bar for secure supply chain practices at important points of industry and community concentration. These policies should also help improve the stability of open-source security efforts and the long-term viability of other recommendations in this report by supporting channels between the public and private sectors. They should also help account for the rapidly growing use of containers in cloud service deployments, including registries and hubs for container and other cloud images.

FIRST

7. **Pay Up** *[US Congress and DHS CISA]***:** Open-source software constitutes core infrastructure for major technology systems and critical software pipelines. The absence of US public support to secure these products, at a cost point far below what is spent annually on other domains of infrastructure security, is a striking lapse. The US Congress should appropriate suitable funds, no less than $25 million annually, and unambiguous grant-making authority, to DHS CISA to support baseline security improvements in open-source security packages through a combination of spot grants up to $500,000 administrated in conjunction with the US Computer Emergency Response Team (US-CERT), and an open rolling grant application process.

8. **Stand and Deliver** *[DHS CISA]***:** In line with the identification of open-source projects as core infrastructure, DHS CISA should create a small (six to eight-person) open-source security evangelism and support organization. This group should help administer funds in the previous recommendation, drive collaboration with US allies and others in the private sector to support priority open-source packages, and act as community liaison/security evangelists for the open-source community across the federal government. The first project for such a group should building a dependency graph of as much of the open source ecosystem as feasible, identifying priority projects for investment and support.

NEXT

9. **Curate and Maintain Tooling** *[CMU's SEI and Linux Foundation]***:** The DoD should create a new nonprofit entity in the public domain

[100] US Department of Commerce, Open Source Code, accessed July 15, 2020, https://www.commerce.gov/about/policies/source-code.

[101] The Center for Strategic and International Studies' Cyber Policy Task Force has similarly recommended that the Trump administration pursue ways to "support open-source software vulnerability research programs, through DHS or perhaps the National Science Foundation." (*From Awareness to Action—A Cybersecurity Agenda for the 45th President*).

[102] New York Cyber Task Force, "Building," 24.

supported by Carnegie Mellon University's Software Engineering Institute (SEI) and Linux Foundation's staff and networks. SEI should support the organization as a principal contractor funded via indefinite delivery/indefinite quantity contract from the DoD, at an amount not less than $10 million annually. The Linux Foundation should manage a priority list of software tools and open-source projects to invest in and support. This entity should support the long-term health of the software supply chain ecosystem by maintaining, improving, and integrating software tools released as part of the Overlay effort, making them freely available with expert integration assistance and other appropriate resources.

10. **Transatlantic Infrastructure Initiative** *[DHS CISA and the State Department]***:** Software security is not a single-jurisdiction issue. DHS CISA and the State Department's Office of the Cyber Coordinator should work with US allies in Europe to establish a Transatlantic Infrastructure Initiative (TII) modeled on the DHS open-source security funding program. Working with ENISA and cooperative partners in the region, the TII could establish a consensus collective security mechanism to support the security of critical open-source packages and help validate the global significance of effective trust in the supply chains for this software.

OVER THE HORIZON

11. **Bring Lawyers, Guns, and Money** *[US Congress/Federal Trade Commission]***:** The US Congress should extend final goods assembler[103] liability to operators of major open-source repositories, container managers, and app stores.[104] These entities play a critical security governance role in administering large collections of open-source code, including packages, libraries, containers, and images. Governance of a repository like GitHub or an app hub like the PlayStore should include enforcing baseline life cycle security practices in line with the NIST Overlay, providing resources for developers to securely sign, distribute, and alert users for updates to their software. This recommendation would create a limited private right of action[105] for entities controlling app stores, hubs, and repositories above a certain size to be determined. The right would provide victims of attacks caused by code, which failed to meet these baseline security practices, a means to pursue damages against repository and hub owners.[106] Damages should be capped at $100,000 per instance and covered entities should include, at minimum, GitHub, Bitbucket, GitLab, and SourceForge, as well as those organizations legally responsible for maintaining container registries and associated hubs, including Docker, OpenShift, Rancher, and Kubernetes.

3.3 Counter Systemic Threats

While there is much progress to be made in providing better tools, incentives, and resources to secure software supply chains, the United States and its allies can also take positive action to call out and counter systemic supply chain threats. This final cluster of recommendations works to broaden existing US partnerships to better reflect a consensus coalition on cybersecurity issues, widen the basis of cooperation to include regular operational collaboration as well as political coordination on law enforcement and intelligence issues, and provide better information to the public on software supply chain risks. Many of these recommendations build on existing relationships and informal partnerships with allies and the private sector.

The abuse of software supply chains undermines trust in the technology ecosystem. Where new or exotic techniques are identified to attack the supply chain, the United States and allies should work to interdict malicious actors and blunt the consequences of these attacks. This section addresses potential actions from shifts in alliance composition to reshaping the focus of interagency equities. The goal of these recommendations is to provide a more concrete basis for statecraft to counter the

[103] Paul Weiss, "The Cyberspace Solarium Commission's Final Report and Recommendations Could Have Implications for Business," *client memorandum*, March 13, 2020, https://www.paulweiss.com/media/3979429/13mar20-cyberspace-solarium.pdf

[104] US Cyberspace Solarium Commission, *A Warning From Tomorrow, March 2020, pp 76-77,* https://drive.google.com/file/d/1ryMCIL_dZ30QyjFqFkkf10MxlXJGT4yv/view.

[105] Trey Herr, "Software Liability Is Just a Starting Point," *Lawfare, April 23, 2020,* https://www.lawfareblog.com/software-liability-just-starting-point.

[106] The New York Cyber Task Force has similarly recommended that "Congress could make it easier to hold software companies liable for products with known, unpatched vulnerabilities and no mature process to identify and fix them."

highest-consequence software supply chain attacks and improve the quality of information available to decision makers and the public on systemic software supply chain threats and the activities of major state actors.

FIRST

12. **Know Your Enemy** *[Office of the Director of National Intelligence]***:** Acting on behalf of the director of national intelligence, the assistant director of the Supply Chain and Cyber Directorate and the national intelligence officer for cyber, with appropriate members of the Intelligence Community and interagency, should produce a study on the nexus between state adversary groups and criminal organizations in any major software supply chain security attack from the past decade. This study should be shared in its entirety with key US allies, including the United Kingdom, Japan, Australia, the Netherlands, Poland, and Estonia, and should include a limited public component released within six months of the study's conclusion.

13. **Trust Busters** *[UK National Cyber Security Centre, DHS CISA, NSA, and DoJ]***:** Building on existing collaboration between CISA, the NSA, and the UK's NCSC, an expanded international group should work to share information on systemic software supply chain attacks and disclosures, cooperate on investigations against responsible groups, and define policies for collective attribution of state actors. NCSC, in particular, offers well-established relationships with telecommunications providers and a strong base of talent to collaborate on such a joint effort.

    a. **Broaden the Tent** *[DHS CISA and NSA]***:** DHS CISA and the NSA should work to include, and routinize with, a broader array of US allies in regular security collaboration, investigation, and joint attribution on software supply chain security events. These allies should include the same states as in recommendation twelve above.

14. **Work for IT** *[US and allied governments]***:** The root of most software supply chain security protections is strong cryptography. Efforts by the United States and other governments to incentivize adoption of weak or compromised cryptographic standards work at cross-purposes with widespread industry practitioner and policy maker consensus. These efforts also present systemic threats to an already porous software supply chain. The legacy of US attempts to block the sale of "strong cryptography" is recurring attacks on weaker encryption standards and widespread access to previously controlled technologies.[107] The United States and its allies should work in support of strong and accessible cryptographic standards and tools for developers.

NEXT

15. **Know Thyself** *[Cybersecurity Tech Accord or Charter of Trust]***:** Industry, working through the Cybersecurity Tech Accord or the Charter of Trust, should annually survey and publish public instances of software supply chain attacks which undermine trust in software updates, code integrity, or distribution channels like public open-source repositories. Each group has an opportunity to assert meaningful leadership on software supply chain security issues with such an effort. These attacks are the evidentiary basis for motivating new investment in supply chain security. Transparency around their frequency and cost are important inputs to public debate.

16. **SBOM Squeeze** *[State Department Cyber Coordinator and Department of Commerce NTIA]***:** The Departments of Commerce and State should collaborate to further internationalize the SBOM effort. Commerce has worked effectively to drive bottom-up engagement, while State should support with a top-down advocacy effort. The transparency associated with SBOM will help surface vulnerabilities and weaknesses which support broader US alliance efforts on cybersecurity. State and Commerce should focus on Germany, the Netherlands, the United Kingdom, Japan, South Korea, and Australia—leading with

[107] Bruce Schneier, *A Worldwide Survey of Encryption Products, Berkman Klein Center for Internet & Society at Harvard University, February 11, 2016,* https://cyber.harvard.edu/publications/2016/encryption_survey.

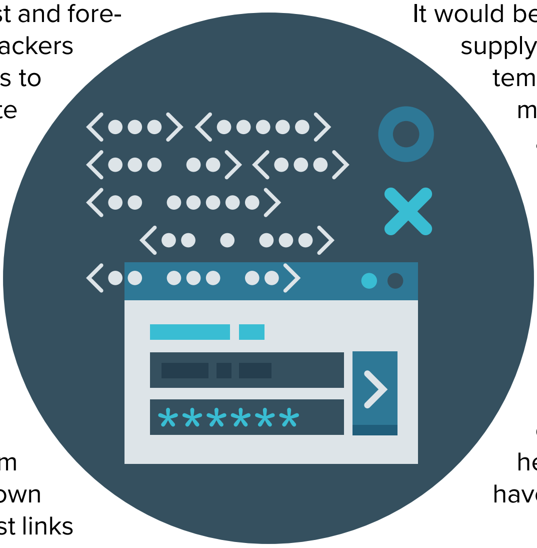
the German Federal Office for Information Security (BSI) and Japanese Ministry of Economy, Trade and Industry (METI) to start.

OVER THE HORIZON

17. **Security for a Common Good** *[NSA, NCSC, other Vulnerabilities Equities Processes]***:** The NSA and the United Kingdom's NCSC should encourage more frequent and fulsome disclosure of known vulnerabilities, or attractive primitives, in key mechanisms of trust for the software supply chain, particularly code signing and means to bypass firmware protections and hardware roots of trust. The process of determining whether to disclose or keep secret an impactful software vulnerability is inherently a tradeoff between different government agencies and as well as divergent conceptions of the public good. This recommendation aims to further tip the scales in favor of disclosure where the subject software vulnerability is principally useful for impersonating legitimate software updates and developer-signed code—the kind whose use, and potential theft or rediscovery, risks further corroding a critical linkage of trust between users and code maintainers.

4. Conclusion

Supply chain security risk is growing and increasingly manifesting as harm to software users. There are many steps between the codebase they first compromise and their final targets, so the distribution vectors of attack—how they ripple across the supply chain—are just as varied as the first point of impact, though the two are often connected. Popular methods of attack include taking advantage of automated updates, compromising software development applications, and sneaking into mobile app stores. Even the act of infiltrating the supply chain with malware is intricate, involving stealing or forging code-signing certificates, breaking into developer accounts, or unearthing hardcoded default passwords.



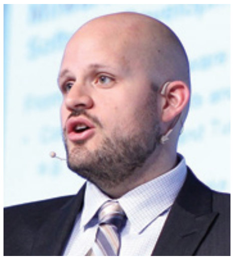
Software supply chain attacks are first and foremost about variety—a variety of attackers ranging from undergraduate students to the world's most sophisticated state offensive cyber groups, of targets that range from uranium enrichment centrifuges to mobile video games, and of impacts that can result in multi-billion-dollar losses, rampant data interception, or absolutely nothing. The supply chains underlying final products grow longer and less linear over time. In this interconnected software environment, successful attacks migrate away from the final targets that harden their own vulnerabilities and toward the weakest links in those chains. The soft spots that software supply chain attacks target remain minimally protected because of the technical challenges of recognizing the full scope of a product's code dependencies and the policy challenges of coordinating disclosure and patching.

A consistent pattern of attacks and disclosures target software supply chains. Despite this, these supply chains remain poorly secured and policy maker attention on supply chain issues is distracted by the 5G debate. This ignores a critical national security risk posed by insecure software supply chains, namely: the accumulated harm to private sector firms impacted by untrustworthy code and a generation of defense systems reliant on commercial software. Software supply chain attacks are popular, they are impactful, and they have been used to great effect by major US adversaries. This report surveyed a decade of software supply chain attacks and disclosures—115 incidents in all—to develop a picture of the problem and develop five trends as software supply chain attacks are used to big effect, break the chain, abuse trust, poison the well, and download trouble. Building on these trends, the report recommends the policy community improve the baseline of software security for all organizations, better protect open source, and counter high-consequence supply chain threats.

It would be a grave mistake to equate software supply chain attacks to a new weapon system in an opponent's arsenal—they are a manifestation of opportunity as much as intent, attacking secure targets by compromising weaknesses in connected neighbors and vendors. Existing gaps in best practices, and poor adoption of these best practices, have granted these software supply chain attacks unnerving sustainability. There are even signs that the most fruitful software supply chain targets in firmware and at the heart of major cloud service providers have yet to peak in popularity.

The implication of this for the technology industry and cybersecurity policymaking community is a crisis in waiting. For the national security establishment, attacks on the software supply chain threaten a generation of technology acquisitions and undermine the COTS model of development. As the recommendations of this report bear out, change is necessary and feasible but it will require concentrated purpose and clarity of outcome at a time when both are in short supply. This report finds evidence that the past decade has seen software supply chain attacks become only more common and effective. Without action, the next decade may be worse.

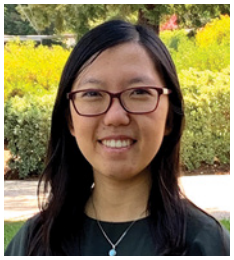
About the Authors



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 the role of the technology industry in geopolitics, cyber conflict, the security of the internet, cyber safety, 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.

William Loomis is a program assistant with the Atlantic Council's Cyber Statecraft Initiative, within the Scowcroft Center for Strategy and Security. In this role, he manages a wide range of projects at the nexus of geopolitics and national security with cyberspace. Prior to joining the Atlantic Council, he worked on market research and strategy at an emerging technology start-up in Madrid, Spain. Originally from New York, he holds a BA in Political Science, with a focus on International Relations and Securities Studies from Colgate University.

Stewart Scott is a program assistant with the Atlantic Council's GeoTech Center. He has a B.A. from Princeton University in Public Policy focusing on misinformation, journalism, and American politics and economic history. He started at the Atlantic Council as an intern with the Cyber Statecraft Initiative.



June Lee is an intern at the Atlantic Council's Cyber Statecraft Initiative, within the Scowcroft Center for Strategy and Security. In this role, she supports a wide range of projects at the nexus of geopolitics and national security with cyberspace. June is also a rising senior at Stanford University, pursuing a BA in international relations and a minor in computer science, with honors in international security studies.

Atlantic Council | Board of Directors

Atlantic Council

The Atlantic Council is a nonpartisan organization that promotes constructive US leadership and engagement in international affairs based on the central role of the Atlantic community in meeting today's global challenges.

Atlantic Council

1030 15th Street, NW, 12th Floor, Washington, DC 20005

(202) 463-7226, www.AtlanticCouncil.org