



Perspectives on Security

Volume One: Securing
Software Supply Chains

A new research series that breaks down emerging security trends and how to address them

Google

Table of Contents

Executive Summary	03	Section 3	17
<hr/>		Industry's SLSA framework can help all organizations securely build and verify the integrity of software	
Foreword	05	<hr/>	
Section 1	09	Section 4	20
Resilience against sophisticated cyber attacks now requires all organizations to secure their software supply chain		A more holistic approach to software supply chain attacks will strengthen defenses world-wide	
<hr/>		<hr/>	
Case Study: SolarWinds	11	Appendix A	24
This case study examines a modern software supply chain attack and its implications		Policy checklist for improving resilience of the software supply chain	
<hr/>		<hr/>	
Section 2	14	Appendix B	25
As open source software use grows, organizations should take on additional security responsibilities to address supply chain risk		Know, Prevent, Fix Approach to Help Secure Software Supply Chains	
<hr/>		<hr/>	
Case Study: Log4j	15		
This case study explores the challenges that organizations face when addressing a serious open source vulnerability			



Modern infrastructure relies on software more than ever

Executive Summary

This month marks the two year anniversary of the [cyber attack on SolarWinds](#), one of the most sophisticated software supply chain attacks in modern history. One key lesson from that attack is the need to work together to better address software supply chain risk. Since the SolarWinds incident, governments have made significant strides in bringing attention to software supply chain issues and accelerated efforts to gain better visibility into the software they procure. Industry has stepped up to support victims and provide technical expertise when attacks take place. At the same time, overall progress across the ecosystem has been slow.

That's why today, we're launching a comprehensive research report with fresh insights to help advance the conversation. This is the first report in a new research series that breaks down the most complex, emerging security trends and explains how Google can help enterprises and governments address them.

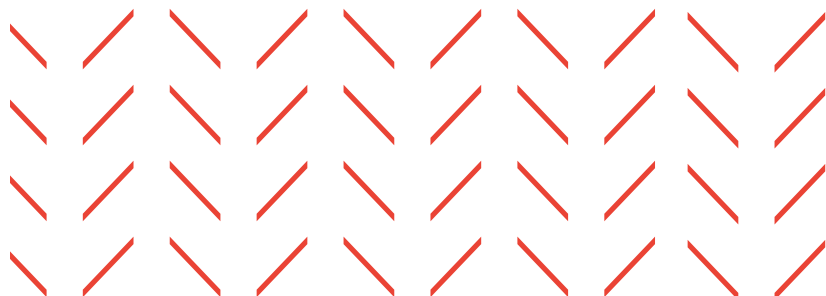


“This is the first report in a new research series that breaks down the most complex, emerging security trends and explains how Google can help enterprises and governments address them”

Importantly, today's report outlines a suggested approach to better understand and address software supply chain risks. And it highlights two key findings.

First, we must work together as a community to [develop and deploy a common Supply-chain Levels for Software Artifacts \(SLSA\) framework](#) that mitigates threats across the entire software supply chain ecosystem. At Google, this has been a requirement for our production environment for almost a decade and we believe the SLSA framework, if implemented properly, would substantially reduce every organization's attack surface.

Second, the lessons we've learned from various security events call for a more holistic approach to strengthen defenses against software supply chain attacks. This includes a common strategy across various stakeholders that centers on three core pillars: 1) adopt best practices and standards for cyber hygiene; 2) build a more resilient software ecosystem; and 3) make investments in the future.



In addition, we share actionable steps that various stakeholders can take right now to help improve software supply chain security.

For Policymakers:

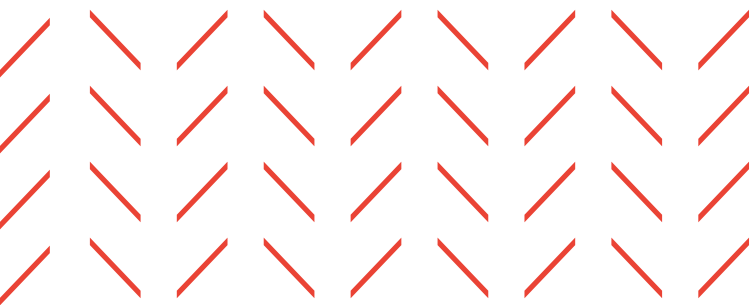
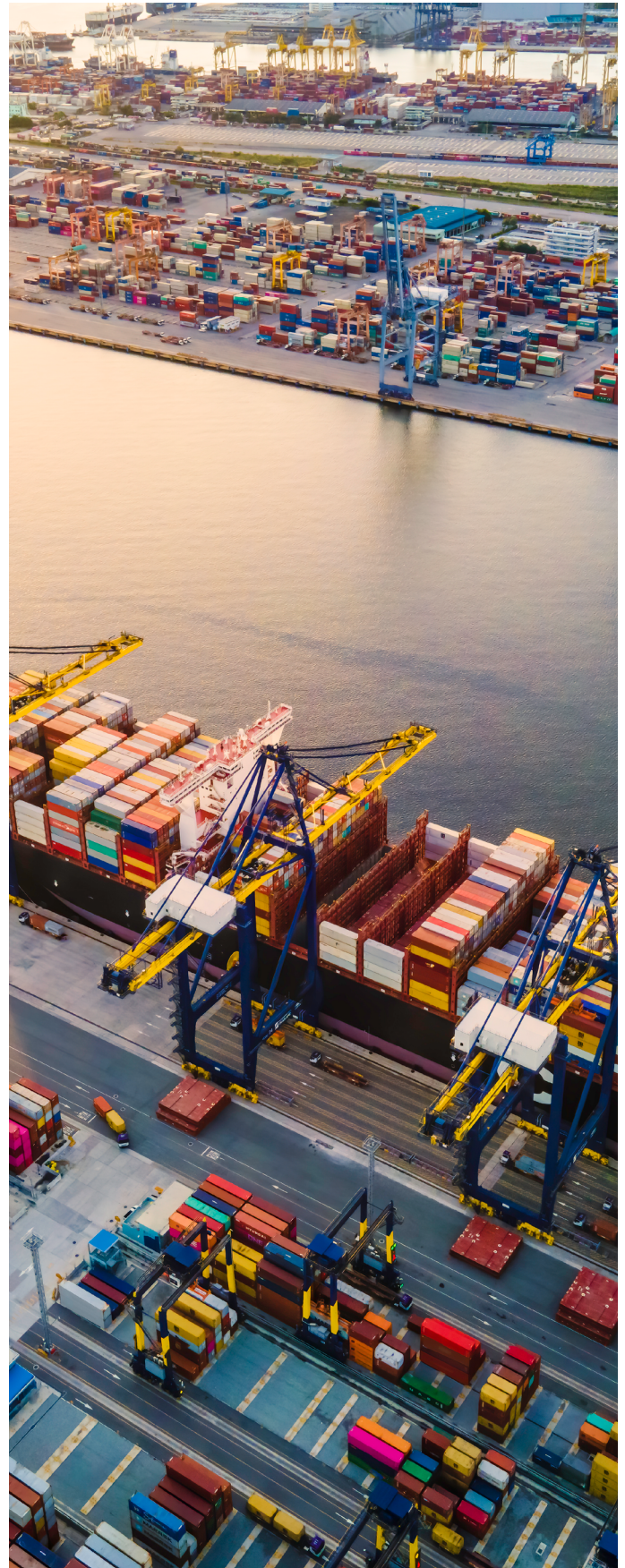
Understand your software supply chain and related infrastructure; develop software supply chain risk mitigation plans; identify potential software supply chain risks; consider security requirements for software procurement; and ensure coordinated vulnerability disclosure plans are in place.

For Practitioners:

Implement [SLSA](#) to harden software supply chain security; cryptographically sign and verify the authenticity of your software using [sigstore.dev](#); automate vulnerability discovery at scale with [OSS-Fuzz](#); automate vulnerability tracking, and triaging with [OSV.dev](#); and automatically evaluate security risk with [Security Scorecards](#).

For Google Cloud Customers:

Equip developers, DevOps, and security teams with the tools they need to build secure cloud applications with [Software Delivery Shield](#); enable access to open source software packages that have been curated and vetted by Google with [Assured OSS](#); explore centralized information about vulnerabilities and possible risks using Google Cloud services like [Security Command Center](#); and leverage best practices like the [Google Cloud security foundation guide](#) to set up more secure authentication and authorization, resource hierarchy, logging, and detective controls.



Foreword

In the summer of 1982, [U.S. satellites reportedly picked up the most monumental non-nuclear explosion and fire ever seen from space](#). The explosion allegedly originated from a Siberian natural gas pipeline and [significantly disrupted the Soviet gas supply and eventually, the Soviet economy](#). Decades later, sources [claimed](#) this incident was no accident.

[It all started with revelations passed to the United States from the French Government](#). According to the reporting, in July of 1981, then French President Francois Mitterrand approached President Reagan on the margins of an economic summit in Ottawa. [President Mitterrand told Reagan that French intelligence had uncovered a longstanding Soviet spying campaign to steal intelligence on Western science and technology to advance Moscow's global ambitions](#). The campaign was unprecedented in scope.

[The French learned about the plot from a source named "Farewell"](#). [Reagan later learned that Farewell was the code name for Colonel Vladimir Vetrov, a 53-year-old engineer in the Soviet Union's intelligence services](#) assigned to evaluate Western science and technology-related intelligence for Soviet spymasters. The Soviets had deployed hundreds of agents worldwide since the 1970s to collect intelligence on advancements in agriculture, radar, computers, machine tools, and semiconductors, among other technologies.

When [President Reagan learned of the spying campaign, he reportedly ordered](#) American national security departments and agencies to introduce modified products and technology into the Soviet supply chain to help contain and reverse Soviet expansionism. In this case, [American intelligence allegedly slipped faulty pipeline software into Soviet procurement channels](#) - software programmed to go

“French intelligence had uncovered a longstanding Soviet spying campaign to steal intelligence on Western science and technology”



French President François Mitterrand met privately with US President Ronald Reagan at the summit in Ottawa, July 1981

haywire and produce unsustainable pressure on joints and welds. The reported result was a [massive five kiloton explosion](#).

To this day, the pipeline explosion precipitated by Farewell's intelligence remains one of most underreported software supply chain incidents in modern history.

Fast forward four decades and the stakes remain high, in part because software powers almost every component of modern society. Today, nearly every organization relies on some piece of software to perform day-to-day operations. Almost all that software is based on internal or external dependencies, open source elements, and includes all the different products, services, people and organizations that go into the software development process. Together, these links form an interconnected software supply chain ecosystem.

Unfortunately, modern day attackers know this and have shown the world that they can exploit those connections to gain access to thousands of systems worldwide. The Russian Foreign Intelligence Service (SVR)'s software supply chain attack on [Solarwinds, for example, targeted the company's popular network management system](#) to slip malicious code into software that was delivered to customers as part of routine updates. The attackers used advanced techniques to mask their work and [operated undetected for months](#). They compromised over 18,000 SolarWinds customers worldwide. In the end, the SVR apparently sought to use their access solely to gather intelligence rather than to disrupt critical networks: they actively exploited only [nine federal agencies and about 100 private sector companies](#).

The sheer numbers of victims, none of whom detected the operation for months, gives some sense of the power of software supply chain operations in a fully connected world and how much more damage attackers in the future could cause. Although the SVR ultimately pursued less than one percent of their opportunities, and, as with the alleged Farewell operation before it, ended up being highly targeted in their activities, future operations handled with less responsibility or greater appetite for risk could see the next Farewell play out on a global scale.

Over the course of the next several years, the frequency and severity of these attacks will likely continue to grow and increasingly focus on open source software—[software with source code that anyone can inspect and contribute enhancements to](#). Open source is a key part of every organization's software supply chain. It is free to use, but costly in terms of manpower to maintain and secure, and it already plays a huge role in critical infrastructure in the U.S. and globally. The emergence of open source software has made software supply chains deeper, wider and more complex: modern-day software tends to include elements published by a large number of diverse open source projects, and often dependencies are several levels deep.

“...the software supply chain is now the second most prevalent initial infection vector into victim systems...”

This creates a massive attack surface ripe for exploitation by nation-states and cyber criminals. Over the past three years, for example, [Sonatype reported that the number of next generation supply chain compromises aimed at actively infiltrating open source software increased by 742 percent](#). From a threat intelligence perspective, [Mandiant recently reported](#) that the software supply chain is now the second most prevalent initial infection vector into victim systems, with seventeen percent of intrusions coming from the software supply chain over the course of 2021—an increase from one percent in 2020. Mandiant attributed that increase to a surge in vendor relationships and a complex matrix of trust relationships. [The average organization now has a unique technology relationship with two hundred and forty-four vendors](#). This translates to thousands of instances of software in use.

As complexity of the software supply chain increases, so does the complexity of responding to vulnerabilities that might occur in any part of the chain – software that incorporates so many dependencies from different sources requires that producers and consumers develop capabilities to rapidly respond when a vulnerability in any one piece of the chain is discovered. When a vulnerability in a widely used package such as Log4j is discovered, mitigation and response efforts can pose significant challenges.

At Google, we believe that software supply chain security is *one of the most critical national security risks facing governments worldwide* and we continue to urge industry, government, and open source community stakeholders to come together to address it. Governments in particular have made important strides in recent months and we're committed to building on that momentum to drive stakeholders towards common goals.

Although software supply chain security has gained a higher degree of visibility in governments and industry in the past few years, at Google we have prioritized this area for far longer. For example, the [SLSA framework](#) discussed later in this report reflects almost a decade of practical experience with designing, building and deploying software supply chain controls that guard the integrity of our internal software supply chain, and govern [admission of software binaries](#) into our production systems.

As leading contributors to the world's software ecosystem, technology companies have a greater responsibility to harden our collective defenses against these attacks and mitigate risks across the ecosystem. Software supply chain resilience requires evolved partnership between the private sector and the public sector to manage shared risks and promote robust collective defense against digital threats.

We take our responsibilities in this area very seriously. That's why [we've committed \\$10 billion to advance cybersecurity](#), and we've pledged [\\$100 million to help enhance open source software security](#). But this is just the beginning. We continue to [create and advance new models for maintaining and securing open source software](#). And we're looking to dramatically expand [our impact](#) with partnerships in government and industry to scale together. Statements are a start, but statements alone won't accomplish the work that needs to be done.

This report outlines a suggested approach to better understand and address software supply chain risks. It also includes the following four recommendations, informed by over two decades of experience managing complex global cybersecurity events.

First, resilience against sophisticated cyber attacks now requires all organizations to secure their open source and proprietary software supply chains. This moves far beyond the headlines surrounding SolarWinds. A growing body of trusted third party organi-

zations have also been targeted, including [Codecov](#) [Kaseya](#), among others. This has led to a dramatic [increase in software supply chain attacks in recent years](#). To make matters worse, clean up is costly. For example, at least one expert has [pegged the cost of containment and cleanup across the ecosystem at one hundred billion dollars](#). [SolarWinds spent eighteen to nineteen million dollars to investigate and remediate the cyber incident](#).

[Software supply chain attacks typically require strong technical aptitude and long-term commitment](#) to pull off. Sophisticated actors are more likely to have both the intent and capability to conduct these types of attacks. Most organizations are vulnerable to software supply chain attacks because attackers take the time to target third-party providers with trusted connections to their customers' networks. They then use that trust to burrow deeper into the networks of their ultimate targets. If defenders stand any chance to stop these malicious actors, we need a better plan to protect vulnerable systems, and it must be scalable. To support this effort, [we've worked with companies across the industry](#) to develop [Minimum Viable Secure Product](#), a vendor-neutral security baseline to help organizations and vendors better align their risk posture.

Second, as open source software use grows, organizations should take on additional security responsibilities to address supply chain risk. The rise in open source software use has triggered a corresponding rise in common dependencies across projects and libraries. [Ninety eight percent of applications use open source software, and open source libraries and components make up more than three quarters of the code](#) in the average software application. Additionally, [the average software application depends on more than 500 open source libraries and components and over seventeen million lines of code](#). The problem isn't getting easier. In 2022, [six out of seven open source project vulnerabilities come from transitive dependencies](#)—meaning attackers can use these common and reliable attack vectors to target a vast number of potential victims.

These dependencies create pressure on organizations to more closely monitor the components in their software ecosystem. They also make it more challenging to engage in best practices like incident response, patch lifecycle and vulnerability management. One way to relieve this pressure is to require Software Bills of Materials (SBOMs) as part of build pipelines for packaged software. [SBOMs are artifacts that provide a nested list of packages and components included in a piece of software.](#) Where consumers are responsible for implementing these security best practices, SBOMs can help them better manage the risks of the software they consume. [SBOMs are supported](#) in the U.S. Administration's efforts to improve software supply chain security, and they complement related frameworks, such as SLSA. For use cases where SBOMs are either not ready for large-scale adoption or a poor operational fit, functional specifications that mandate security best practices can help improve supply chain security.

Third, industry's SLSA framework can help all organizations securely build and verify the integrity of software. [SLSA is an open source framework for software supply chain security](#) that includes standardized vocabulary and a checklist of controls and practices to prevent tampering, improve integrity, and secure packages and infrastructure. [If an SBOM is like an ingredient list on a food product in the grocery store, then SLSA can be thought of as all the food safety handling guidelines that ensure the final product is safe for consumption.](#) Put another way, SBOMs are most helpful when used alongside tools like SLSA for managing software supply chain risk. Organizations which consume, build, or operate software can use the SLSA framework to apply policy governing how artifacts are deployed, executed, and incorporated into other software products. We believe that using this framework could have helped organizations prevent or reduce the impact of major supply chain events like those described in this report.

Finally, the lessons we've learned from various security events call for a more holistic approach to strengthen defenses against software supply chain attacks. This includes a common strategy across government, industry, academia, and the open source community to better equip all stakeholders with the tools they need to address software supply chain risk. Consistent with ongoing software supply chain work across government and industry, the strategy must center on three core pillars: 1) adopt best practices and standards for security hygiene; 2) build a more resilient software ecosystem; and 3) make investments in the future. Working across all three pillars, we can both prepare for—and respond to—future attacks.

Overall, this approach is rooted in a basic principle: we defend better together. We hope this report serves as a call to action for everyone to do more to prevent these attacks. At Google, we are committed to doing our part to support these efforts and look forward to partnering with others to drive continued progress and help organizations, businesses, governments, and users stay safe online.

“[We need] a common strategy across government, industry, academia, and the open source community to better equip all stakeholders with the tools they need to address software supply chain risk”

Section 1:

Resilience against sophisticated cyber attacks now requires all organizations to secure their software supply chains

Civilizations have always relied on complex supply chains to deliver goods and services across geographies to large populations. During the peak of the Roman Empire, for example, the [Roman government created one of the world's most complex end-to-end supply chain operations—one ship per hour every day of the year—to deliver 320,000 free bread loaves per month to citizens of Rome](#). Centuries later, in World War II, Allied Powers built an [extensive support network to supply and maintain large military forces](#) and deliver tanks, airplanes, food and aid to the frontlines. That system [played a decisive role in winning the war](#).

Modern supply chains have since evolved to include [bigger containers](#), [better integration](#), and [faster delivery](#). But today, we face a different challenge. [Software is now an integral part of every industry, not just the software or tech industries](#), and [open-source components now comprise 90 percent of most software applications](#). Almost all that software is based on internal or external dependencies, including all the different products, services, people and organizations that go into the software development process. Together, these links form an interconnected software supply chain ecosystem. And that system is vulnerable.

Indeed, nation-state sponsored malicious cyber actors and cyber criminals will increasingly seek to exploit vulnerabilities in software supply chains to compromise sensitive systems worldwide.

We've seen a surge of activity in the past three years, leading to a significant [increase in software supply chain attacks](#). [These attacks impacted almost every sector and included hundreds of millions of victims](#).

The View from Google's Threat Analysis Group



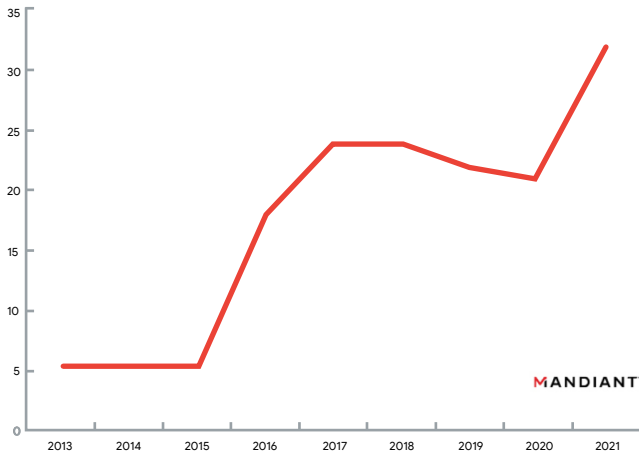
[Google's Threat Analysis Group \(TAG\)](#) counters threats from government-backed attackers and cybercrime groups to the company and its users. Over the past several years, TAG has observed various threat actor groups attempt to compromise software supply chains to engage in malicious activity, such as [mining bitcoin, and stealing cryptocurrency and credentials](#). One recent example of this involves activity with Python Package Index (PyPI), a repository of software for the Python programming language. As of the drafting of this report, [PyPI has over 639,174 active users, hosts 415,798 projects, with 3,938,228 releases](#).

In August, TAG observed attackers "[typosquatting](#)" the names of legitimate packages in PyPI to distribute malware to a large number of victims. We've seen this technique commonly used in supply chain attacks and used as far back as 2017 to compromise [multiple components of production software](#). In this case, cyber criminals created a phishing site [that mimicked PyPI's login page](#). Criminals used the login page to steal credentials and then use that information to create legitimate looking, but malicious packages that would execute malicious code anytime the package was installed from PyPI.

The compromised PyPI package targeted Windows users only and fetched and executed a .NET written malware to install malicious extensions in popular browsers such as Firefox, Chrome and Edge. The malware then exfiltrated any saved credentials from the browser, with all information then sent to an attacker-controlled domain. TAG worked with others in the community to minimize potential impact to users and these malicious package releases have since been removed from PyPI.

Supply Chain Compromises

2013 - 2021



Most state-sponsored supply chain compromises that Mandiant identified in 2021 were linked to China. Continuing a [trend](#) from 2019 and 2020, Chinese threat groups implanted malicious code into government software, including a [biometric fingerprint scanning program](#) used by a South Asian government and the Mongolian certificate authority [MonPass](#), likely to support intelligence collection efforts in those countries. In 2021, Mandiant observed financially motivated actors [FIN7](#) and [UNC2465](#)—which we typically associate with ransomware deployment—target software supply chains to compromise systems. Threat actors also used this tactic to [deliver](#) cryptocurrency miners and banking trojans. Separate reporting described how threat actors used tainted packages and mobile applications to facilitate cryptocurrency theft and mining, ad fraud, and scams related to service subscriptions.

A software supply chain attack [occurs when someone infiltrates an organization's system through an outside partner or provider with trusted access to its systems and data](#). This tactic is attractive for attackers because they can compromise one trusted vendor and through that access, [distribute malicious code to thousands, if not millions, of victims](#). With a single breach, for example, attackers can [read sensitive emails](#), [steal source code](#), and even [operate inside victim organizations undetected for months](#).

But it's not just the volume of attacks that's concerning. Attackers have demonstrated they can target software supply chains to gain access to some of the [most well known](#)—and seemingly well protected—organizations in the world. It's helpful, then, to recall a phrase attributed to Scottish philosopher Thomas Reid, “the chain is only as strong as the weakest link, for if that fails the chain fails”. This means these attacks threaten everyone and it's incumbent on the entire community [to work together to better secure digital infrastructure worldwide](#).

It is important then to pause and reflect on the two year anniversary of the SolarWinds supply chain incident. This event both defines a modern software supply chain attack and offers a case study in the security community's response to such an attack.



SolarWinds Corp. headquartered in Austin, Texas, disclosed a major cyber attack in 2020

Case study: SolarWinds

“These claims are like a bad detective novel ... but [SVR] would be flattered if [it] had been responsible for such a sophisticated attack”

*Sergei Naryshkin, Director,
Russia's Foreign Intelligence Service | [Source](#)*

On February 17, the White House [announced that malicious cyber actors had launched a broad and indiscriminate effort](#) to compromise the ubiquitous [SolarWinds Orion product](#)—a popular network management tool used by hundreds of public and private sector organizations. The attackers used that access to compromise [nine federal agencies and about 100 private sector companies](#), representing the most significant nation-state software supply chain attack in modern history.

The revelation capped off months of investigative work across the public and private sectors. During that time, investigators made several startling discoveries.

First, the attackers targeted a popular network management tool that they likely knew could then help them obtain access to the networks and systems of multiple high value targets. Second, the attackers used sophisticated tactics, techniques, and procedures to operate undetected across victim organizations for months. This included methods to counter security tools and forensic examination and efforts to tailor their capabilities specifically to certain targets. Third, the attackers gained deep and persistent access to sensitive information, including private sector source code and government emails. Finally, the attackers demonstrated both restraint and focus in their efforts to target specific information and specific people. They were selective.

Taken together, this evidence pointed toward a sophisticated nation state actor.

Indeed, in April, the U.S. Government (alongside numerous like-minded allies) publicly [attributed the attack to the SVR and imposed sanctions on the Russian Government](#) in response. Over the past three decades, the SVR was responsible for recruiting some of the most high profile secret agents in American history, including [Aldrich Ames](#) and [Robert Hanssen](#). Today, [the SVR remains actively engaged in intelligence operations](#), and the SolarWinds incident is consistent with the organization's long standing mission to conduct

What do we know about the SVR's digital operations?



Emblem of the Foreign Intelligence Service of the Russian Federation and graphical representation of APT29, a Russia-based espionage group

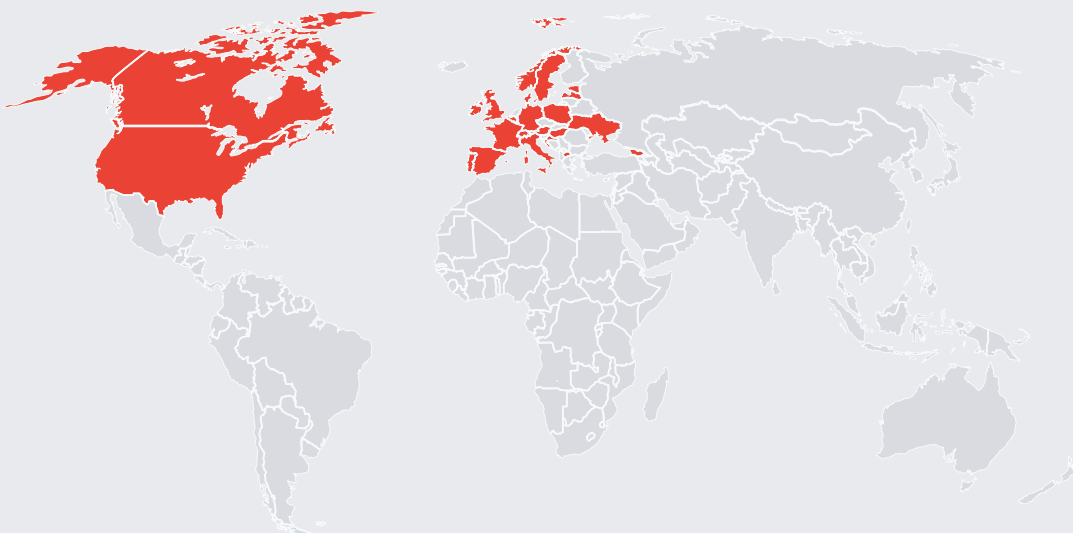
[Since 2014, Mandiant has been tracking APT29, a Russia-based espionage group assessed to be sponsored by the SVR.](#) During that time, APT29 has continued to advance its significant technical trade-craft and operational security, including consistent and steady advancement in tactics, techniques, and procedures. APT29 has used those capabilities to target a range of industries worldwide.

Initial Infection Vectors Used by APT29

	2014	2015	2016	2017	2018	2019	2020	2021	2022
Stolen Credentials	●						●	●	
Phishing Email	●	●	●		●			●	●
Server Compromise		●							
Password Spray					●	●	●		
Supply Chain Compromise							●		
Additional Third Party							●	●	

APT29 has used those capabilities to target a range of industries worldwide

APT29 Victimology



- Biotechnology
- Consulting
- Education
- Financial Services
- Government
- Healthcare
- Legal Services
- Nonprofits
- Pharmaceuticals
- Technology
- Telecommunications
- Think Tanks
- Travel
- Science/R&D

[industrial espionage to support Russian domestic technology development and national defense.](#)

The investigation into the SolarWinds incident revealed several gaps in the U.S. private and public sector's collective effort to detect and respond to software supply chain incidents. First, the attackers launched the attack from inside the United States. This, combined with a lack of domestic incident reporting requirements to appropriate federal authorities, [made it difficult for the US Government](#) to connect the dots, instead relying on victims in the private sector to voluntarily step forward. Second, the US Government lacked effective internal coordination in working with the private sector on incident response of this magnitude. This likely [impeded quick decision making and information sharing](#) internally as related to impacted government systems. Finally, the federal government didn't have a broader strategy for addressing software supply chain risks.

The President addressed many of these gaps in May 2021 with [Executive Order 14028, Improving the](#)



D/NSA Neuberger updated the nation on cybersecurity issues at the White House in Washington, D.C., on Monday, March 21, 2022

[Nation's Cybersecurity](#) ("the E.O."). To improve information sharing and incident response, the E.O. directed various agencies and departments to take action, including among other measures: 1) establish a [Cyber Safety Review Board](#) to review and assess significant cyber incidents; 2) develop a standard set of operational procedures or playbook to help plan for and respond to cybersecurity vulnerabilities and incidents; and 3) remove contractual barriers to information sharing from federal contractors. [The Cybersecurity and Infrastructure Security Agency \(CISA\) has since published playbooks](#) to help agencies and departments manage incident and vulnerability response. Congress has since [passed legislation](#) that empowered [CISA to develop and implement regulations](#) for mandatory private sector reporting of certain cyber incidents and mandatory reporting of ransomware payments.

To advance software supply chain security, the E.O. directed the National Institute of Standards and Technology (NIST) to [publish guidelines that include criteria to evaluate the security practices of developers and suppliers of software](#), and guidance that identifies practices that enhance the security of the software supply chain. NIST later issued [guidelines on the recommended minimum standards for vendors' testing of their software source code](#) and guidance outlining [security measures for critical software use](#). The E.O. also introduced new procurement requirements—updating Federal Acquisition Regulations, for example—to [leverage the government's purchasing power to improve baseline security practices for all federal agencies and departments](#). The White House continues to use this and [other mechanisms to advance strong software supply chain policies](#).

But software supply chain attacks aren't just defined by events like SolarWinds. They raise broader questions about trends in secure software development and the systems we depend on to find and fix major vulnerabilities in the Internet ecosystem.

Section 2:

As open source software use grows, organizations should take on additional security responsibilities to address supply chain risk

Studies in modern ecological systems reveal certain [keystone species](#). A keystone species helps define an entire ecosystem and can include any organism from plants to fungi. [Without its keystone species, the ecosystem would be dramatically different or cease to exist altogether](#). There is no greater 'keystone species' parallel in the Internet ecosystem than open source software—as a global community, [we've become highly dependent on its use](#). In 2022, for example, [the number of open source dependencies being downloaded and integrated into software grew by an estimated thirty-three percent](#). Open source software is ubiquitous and used to power many popular products and services like [operating systems \(Linux\)](#), [databases \(MySQL\)](#), and [websites and blogging platforms \(WordPress\)](#).

Open source software is not uniquely risky, however, all software requires lifecycle management to ensure its security. This poses challenges for consumers of open source software when it comes to best practices like incident response and vulnerability management. Most of the current work to enhance open source software security, for example, is done on a voluntary basis. This means that organizations take on a host of security responsibilities to address supply chain risk, including assessing the quality of dependencies they consume (i.e., is this enterprise-grade or a hobby project) and ensuring they have the right mechanisms to receive and ingest new information on vulnerabilities present in the open source software they use.



Pisaster ochraceus sea stars like this one were the first animals to be identified as keystone species

In addition, the fundamental nature of open source software—that anyone is free to download and use it, including by integrating it into another piece of software—means that there is no easy way to systematically and comprehensively remediate vulnerabilities. Once a vulnerability surfaces, for example, security researchers may not be able to directly contact all users of the software to ensure that the users know about a newly discovered vulnerability, or the appropriate steps to remediate.

This presents obvious hurdles when incidents happen. The Log4j vulnerability—which set off one of the most aggressive global incident response efforts to date—demonstrates how far we have to go to secure open source software. And it highlights the need to help users understand that they have a responsibility to confirm if an open source vulnerability like Log4j impacts them and whether they need to take additional steps like patch their systems.

Case study: Log4j

“The Log4j vulnerability is the most serious vulnerability I’ve seen in my decades-long career”

*Jen Easterly, Director,
Cybersecurity and Infrastructure Security Agency | [Source](#)*

Almost one year after the SolarWinds event, researchers publicly disclosed [a critical vulnerability in Log4j](#). Log4j is an open-source Java logging library developed by the Apache Foundation. [It is widely used in many applications and is present in many services as a dependency. This includes enterprise applications, including custom applications developed within an organization, as well as numerous cloud services.](#) NIST [quickly assigned the security vulnerability in Log4j the maximum possible score](#) –meaning it posed a grave risk to organizations and needed to be fixed immediately.

The Log4j vulnerability raised two critical issues for the security community.

























First, Log4j was simple to exploit, exploitation could be conducted remotely, and the code to do so was public and easy to automate. This created opportunities for mass exploitation across the Internet. As a result, once the Log4j vulnerability was publicly disclosed, attackers seized on the information to attempt [to exploit more than forty-four percent of corporate networks worldwide](#).

Second, Log4j revealed weaknesses in how large organizations track their own software deployments and map associated dependencies. Without this information, defenders can’t find and fix software bugs quickly, which meant that in many organizations, it was unclear where to even start their remediation efforts or how large the effort would be.

In responding to Log4j, Google wore several hats.

First, we managed potential impact to Google systems. [Google’s analysis of Log4j](#) found that more than eighty percent of the packages we observed in the wild had a vulnerable dependency more than one level deep, with a majority affected five levels down (and some as many as nine levels down). This information helped inform and accelerate our [incident response process](#), including activating and driving remediation across all of the company’s business units. To help inform first responders, we published a [detailed, platform-agnostic blog](#) that assessed the impact of the vulnerability across the ecosystem.

Following public disclosure, Google observed scanning for vulnerable instances of Log4j, although we weren’t able to confirm whether this scanning was malicious activity. The scanning activity began very shortly after the vulnerability became public and continued over the following weeks. One month after initial disclosure, Mandiant [reported](#) that it was tracking more than twenty-two clusters of threat activity across the global ecosystem exploiting the vulnerability, including suspected Chinese and Iranian state-sponsored threat clusters APT41, UNC3500, and UNC2448.

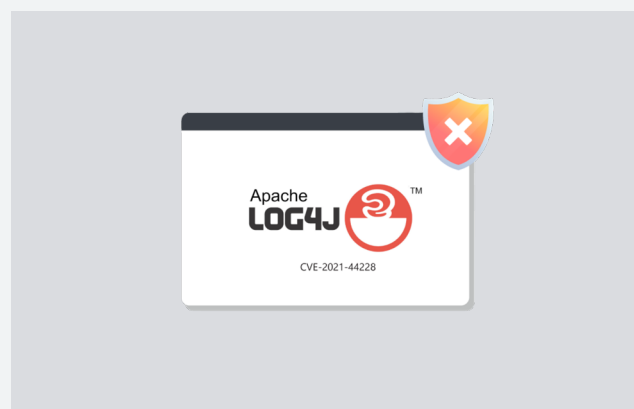
Assessed Motivation	Threat Groups	Targeted Industries
 Cyber Espionage	APT41 UNC2448 UNC3500	 Financial  Government  Higher Education  Telecommunication
 Cyber Crime	UNC961 UNC3007 UNC3543 UNC3569 UNC3581 UNC3594	 Business Services  Financial  Food Services  Government  Healthcare  Higher Education  Media  Retail  Transportation
 Unknown	UNC3510 UNC3581 UNC3614	 Business Services  Financial  Food Services  Government  Higher Education  Retail  Technology  Telecommunication

[in the United States and beyond](#). This included sharing information on what we were doing to respond to the vulnerability and recommendations on how to fix it. We also played a lead role in supporting the government’s efforts to learn from Log4j and prepare for the next major event. This included briefing the Cyber Safety Review Board—[the organization tasked with writing a comprehensive report on the collective efforts to respond to Log4j](#)—on Google’s overall response. Those insights were reflected in the [final Log4j report](#).

But our work is far from done. Log4j is [an endemic vulnerability and likely will remain present in systems for years to come](#). Through continued partnership, we can improve our collective ability to respond to events like Log4j and help organizations better prepare for the next one. And that preparation starts now, with a more deliberate effort to encourage organizations to move forward with the SLSA framework.

Second, as a leading cloud service provider, we quickly took steps to protect Google Cloud customers. This included [publishing technical information and associated recommendations to help customers respond to the vulnerability](#). We also developed [new tools](#) to help them detect and, optionally, block commonly attempted exploits while they were working on patching their systems.

Finally, we leaned forward with governments to help drive better public and private sector collaboration on Log4j. Through the [Joint Cyber Defense Collaborative](#), for example, we [worked with the government and industry partners to support the broader response effort](#)



Log4j Community logo

Section 3:

Industry's SLSA framework can help all organizations securely build and verify the integrity of software

SolarWinds and Log4j represent two of the most significant software supply chain events in recent years and it's important to learn and adapt our approach to future events. But focusing on solutions to those events alone will obscure a range of threats across the software supply chain ecosystem. Attacks like [Codecov](#) and [Kaseya](#), for example, targeted completely different components of the software supply chain.

There is now an urgent need to develop and implement a comprehensive end-to-end framework that defines how to mitigate threats across the entire software supply chain and provides some guarantee that following that framework will improve security for everyone. [We've been working with others in the community](#) to develop such a framework for more than a year and at Google, it's been a requirement for our production environment for almost a decade. We believe the framework, if implemented properly, would substantially reduce every organization's attack surface.

We call this solution [SLSA](#). SLSA is an open source framework for software supply chain security that includes standardized vocabulary and a checklist of controls and practices to prevent tampering, improve integrity, and secure packages and infrastructure. Organizations which consume, build, or operate software can use the SLSA framework to apply policy governing how artifacts are deployed, executed, and incorporated into other software products.

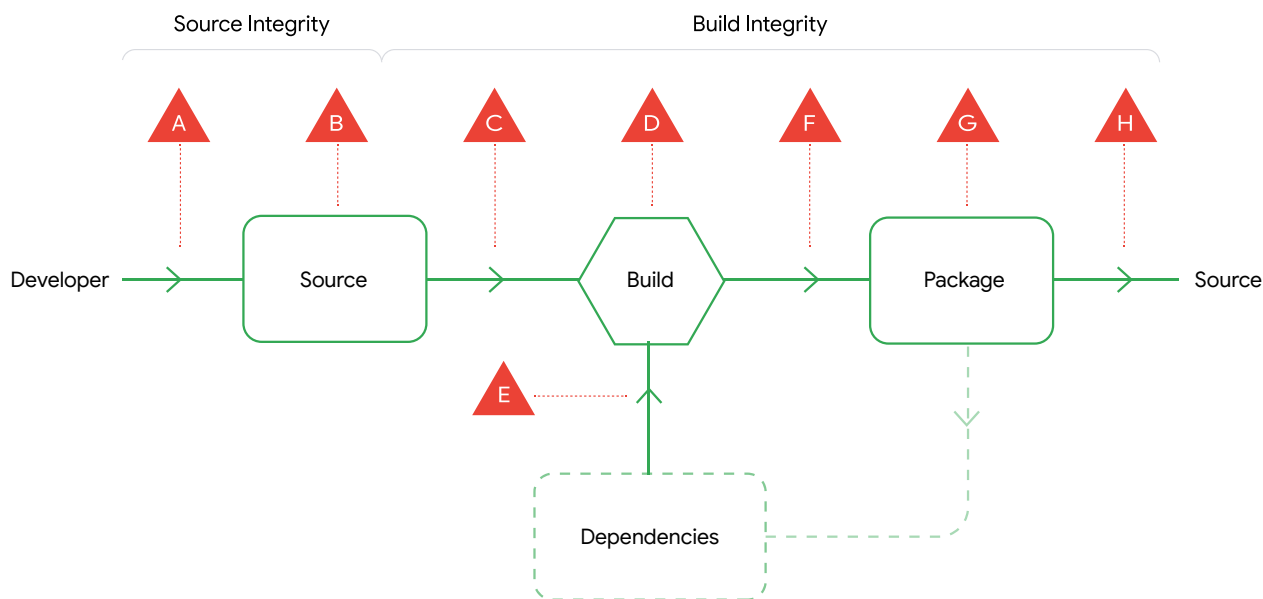
In addition to Google, organizations like [VMWare](#), [Red Hat](#) and [Suse](#) have implemented the SLSA framework. And an [increasing number of organizations are adopting at least some of these emerging security practices](#) to improve software security.

For implementers, a key aspect of SLSA is its [leveled structure](#). This is a deliberate choice to increase its useability: If you think about SLSA levels as the rungs of a ladder, it's easy to get a grip on the bottom rung; and wherever you are on the ladder, the next rung is always within reach. SLSA has the pragmatic flavor of a transformation framework, recognizing that adoption is a multi-step journey for most organizations. And SLSA is designed to evolve as its adoption grows to address future use cases.

SLSA addresses common attack vectors in a typical software supply chain threat model. They include: submission of bad code; compromise of source control; modification of code; compromise of build platform; use of bad dependency; bypass of CI/CD; compromise of package manager; and use of bad packages.



Supply-chain threat model



- A** Submit unauthorized change
- B** Compromise source repo
- C** Build from modified source
- D** Compromise build process
- E** Use compromised dependency
- F** Upload modified package
- G** Compromise package repo
- H** Use compromised package

To make this framework more actionable, we’ve mapped these eight attack vectors to recent software supply chain attacks. We’ve also included a brief description of how SLSA could have addressed them.

Threat	Known Example	How SLSA Could Have Helped
A Submit unauthorized change (to source repo)	SushiSwap	Two-person review could have caught the unauthorized change
B Compromise source repo	PHP	A better-protected source code platform could have been a harder target for the attackers
C Build from modified source (not matching source repo)	Webmin	A SLSA-compliant build server would have produced provenance to identify sources used and detect tampering
D Compromise build process	SolarWinds	Higher SLSA levels require stronger security controls for build platform, making it more difficult to compromise and gain persistence
E Use compromised dependency (i.e. A-H, recursively)	Event-stream	Provenance would have indicated that it wasn’t built from a proper builder or that the source did not come from GitHub
F Upload modified package (not matching build process)	CodeCov	Provenance would have indicated that the artifact was not built as expected or from the expected source repo
G Compromise package repo	Attacks on Package Mirrors	Provenance would have indicated that the artifacts were not built as expected or from the expected source repo
H Use compromised package	Browserify typosquatting	Provenance linking back to source control can enable and enhance other solutions

Anticipating the industry need for a standard supply chain integrity framework, Google open-sourced SLSA, which was used internally in Google's production workloads. **SLSA is now playing an increasingly central role in addressing the government's larger software supply chain security concerns in 2022 and beyond.** Industry and open source stakeholders recently pointed to SLSA as one of the most important tools to help advance the state of software supply chain at the [Open Source Software Security Summit](#) in February, and in October the [US Government referenced SLSA in a report on recommended practices to help suppliers secure their software supply chains](#). And, we're working with the open source community to aggressively [expand SLSA capabilities to help guide practitioners towards a workable solution](#) to NIST's newly published [Secure Software Development Framework](#) guidelines. This is critical because [the federal government will soon require agencies to comply with these guidelines](#) and all software vendors will need to adhere to them to sell software to the government in the future.

SLSA also complements broader software supply chain security initiatives worldwide. The European Union Agency for Cybersecurity (ENISA), for example, [highlighted SLSA's contributions to helping organizations defend against supply chain attacks](#). We continue to support this work.

What these useful efforts have so far failed to do, however, is to stitch together all the work the community is doing to make progress on software supply chain security and connect it across multiple geographies. We urge global governments to pursue alignment on these issues to the greatest extent possible to avoid fragmentation and adoption of measures that would stifle innovation. In the longer term, we need the public and private sectors to come together to develop a more holistic approach to strengthen defenses worldwide.



Preparations start now to move forward with the SLSA framework

“[W]e’re working with the open source community to aggressively expand SLSA capabilities to help guide practitioners towards a workable solution to [SSDF] guidelines”

Section 4:

A more holistic approach to software supply chain attacks will strengthen defenses worldwide

We'll soon reach a milestone in the security community: **the two year anniversary of the SolarWinds attack**. This presents an opportunity to pause and reflect on the progress we've made. Governments have made significant strides in bringing attention to software supply chain issues and accelerated efforts to gain better visibility into the software they procure. And industry has stepped up to support victims and provide technical expertise when attacks take place. But overall progress across the ecosystem has been slow, and we'll need a much more holistic approach to software supply chain attacks to strengthen defenses worldwide.

This includes a common strategy across government, industry, academia, and the open source community to better equip all stakeholders with the tools they need to address software supply chain risk. Consistent with recommendations we supported with the [Cyber Safety Review Board](#) and made in other government and industry forums, the strategy must center on three core pillars: 1) adopting best practices and standards for security hygiene; 2) building a more resilient software ecosystem; and 3) making investments in the future. Working across all three pillars, we can both prepare for—and respond to—future attacks.

Before jumping into the pillars, we should first recognize that **legacy infrastructure is imposing unacceptable security costs across the Internet ecosystem**. Governments in particular continue to rely on technology providers to provide solutions to the same security problems they create. Malicious actors know this, and they continue to target the same legacy vendors and

exploit the same common vulnerabilities to target victims. To fix this, governments should seek to diversify their procurement strategies and leverage procurement processes to hold legacy vendors accountable for poor security practices.

In contrast, [IT modernization](#)—in particular through **the embrace of cloud technology**—can dramatically improve security. Economies of scale enable cloud providers to reduce the costs of providing state-of-the-art security capabilities for organizations of all sizes. Cloud providers are able to gather intelligence on threats and techniques from thousands of customers and share lessons learned with the community, creating positive feedback loops for security. At a basic level, this drive towards modernization will raise the security bar for everyone because it enables more organizations to adopt controls that are considered out of reach for many today, such as a comprehensive [zero trust architecture](#) and strong hardware authentication.

Best practices and standards for security hygiene

In Bill Walton's autobiography, [he recalled his first practice with legendary basketball coach John Wooden. In his telling of the story, the young recruits eagerly sat down to hear wise words of wisdom from the legend](#). John sat down on a stool and began his first lecture, "this is how you put your shoes and socks on." Bill and others were shocked. They expected something much

more prophetic, but would soon learn John was teaching them a cardinal rule about life: to be successful, you have to practice the fundamentals well.

In security, the same principle applies. Organizations should [focus on the basics](#). Indeed, [passwords are still the root cause of over eighty percent of breaches](#) and [the single most important thing that users can do to stay safe online is enable multi-factor authentication \(MFA\)](#). At Google, we support this work. We were the first consumer technology company to automatically turn on two step verification, our version of MFA for our users. And we work with other industry partners in the [FIDO Alliance](#) to advance technologies to stay ahead of future threats, including creating a phishing-resistant form of MFA.

Through that effort, [we've also been able to bake FIDO protocols into operating systems, browsers, phones, and tablets](#). To build better resiliency across the Internet, we also support the US Government's recommendation that every organization [include FIDO authentication on its MFA implementation roadmap](#).

Our work on two step verification reflects our broader approach to security with [multiple layers of defense](#)—we call this **defense in depth**—to help protect against attacks. This helps us create an IT infrastructure that is more secure and easier to manage than more traditional technologies and at the same time, continue to push the boundaries with advanced state-of-the-art capabilities. To improve the security of the Python ecosystem, for example, [we provided security keys to developers of critical python projects](#) to ensure that maintainers of critical projects have the ability to implement strong MFA.

At Google, [security is the cornerstone of our product strategy](#), and we've spent the last decade building [best-in-class infrastructure](#) and designing products that implement security at scale: every day, for example, [Gmail blocks more than 100 million phishing attempts](#).

We bake in security from the beginning instead of bolting it on as an afterthought and we design helpful products that are secure by default for our users. We were the first large organization to implement zero trust at scale, and the first cloud provider to turn encryption on for all data-at-rest by default to not only provide strong protection for customer data but also to promote widespread adoption of best practices.

We also partner closely with industry stakeholders to identify and address vulnerabilities in the open source software ecosystem, and share best practices on how to address the latest security threats. To help others, for example, we [proposed a step-by-step process to address open source vulnerabilities](#) (for more information on this process, see Appendix B). In addition, Google contributed heavily to Open Source Security Foundation's (OpenSSF) [guide on coordinated vulnerability disclosure \(CVD\) for open source projects](#). The guide was based in part on [Google's prior CVD publication](#) and suggests a process for publicly disclosing vulnerabilities, and includes commonly-needed policy and communication templates, such as embargo notifications and disclosure announcements.

In addition, in partnership with [OpenSSF](#), Google helped establish [Security Scorecards for Open Source](#)—an effort to automate evaluation of security best practices for critical open source projects. We've made this data public for one million critical open source projects across various software languages. This work also aligns with existing [Secure Software Development Framework](#) (SSDF) requirements.

Working together, we can encourage the security community to embrace these best practices and standards and help provide a roadmap for better security outcomes.

Building a more resilient software ecosystem

For almost two decades, Google has been [an industry leader in building better software and driving open source innovation across the ecosystem](#). Open source contributors at Google work on a variety of projects and repositories—not just our own code. We sponsor, create, and invest in projects and programs that enable everyone to join and contribute to the global open source ecosystem. We will continue to make open source security a priority and urge others to do the same, because the health and availability of open source projects strengthens the security posture of users and developers everywhere.

For example, in 2021, we launched [Open Source Insights](#), a tool designed to list and visualize a project's dependencies and their properties. When Log4j broke in late December 2021, the [Open Source Insights](#) team reported that over 35,000 Java packages were impacted by the vulnerability and compiled a [list of 500 affected packages](#) to help guide patching and remediation activities.

Still, there is more to do. As of July, the [Open Source Insights](#) team reported that [only 40% of the affected packages have remediated the problem](#). To support enterprise and public sector customers, we've launched [Assured Open Source Software service](#) to help them reduce their need to develop, maintain, and operate complex processes to secure their open source dependencies. We've also launched [Software Delivery Shield](#), a fully managed software supply chain security solution to help equip developers, DevOps, and security teams with the tools they need to build secure cloud applications.

We continue to point out that truly moving the industry forward requires embedding controls throughout the

build process. SBOMs provide point-in-time view of software composition, offer greater transparency into software dependencies, and help software consumers better execute critical security tasks, like patch lifecycle management, vulnerability management, and incident response. But SBOMs don't address the full scope of risks in the supply chain. For example, SBOMs may be useful for self-hosted offerings where the consumer is responsible for implementing security best practices. However, SBOMs would be more difficult to implement and less useful for SaaS offerings because SaaS software is dynamic and components vary. In these cases, it might be more helpful to enhance functional requirements in programs like FedRAMP to mandate best practices for vendors in areas such as vulnerability management.

Together, a combination of [SBOMs, functional specifications, and complementary security frameworks and tools can help reduce software supply chain risk](#). That's why we worked with [OpenSSF](#) to publish the SLSA framework and continue to follow up with best practices, standards, and relevant products and services to help implement it. [We also partnered with Red Hat, Purdue University, and others to create sigstore](#). Similar to the [Let's Encrypt](#) effort for Secure Sockets Layer/Transport Layer Security, sigstore is designed to enable widespread adoption of digital signatures across the software supply chain.

More broadly, Google has a dedicated team of full-time engineers focused on creating innovative security solutions to improve open source security. In partnership with OpenSSF, for example, we've founded projects like [Open Source Vulnerability schema](#). This helped several ecosystems align on a consistent standard for communicating about known vulnerabilities, and resulted in [a comprehensive vulnerability database](#) that was open, distributed, and precise.

Google has also continued to provide free continuous fuzzing service [OSS-Fuzz](#) to the open source commu-

nity. Together, we've onboarded over [seven hundred projects and reported and fixed eight thousand vulnerabilities](#). We are also continuing our efforts to advance the state of the art for fuzzing, through our continued collaborations with academia and efforts to automatically [find more classes of serious vulnerabilities](#). [We've also created a new Open Source Maintenance Crew to engage upstream full time](#) to expand this and related work across critical open source projects.

While this paper focuses primarily on the risks associated with open source software, it is also important to highlight the benefits, including its capacity to enable stakeholders to collaborate on security improvements. For example, [VS Code](#) is one of the [most popular integrated development environments \(IDEs\)](#) used by developers to edit source code, including at Google. As part of efforts to improve the security of web-based applications, our security engineers developed [techniques](#) to [prevent vulnerabilities](#) in web apps, and collaborated on [changes to the VS Code](#) open source project to take advantage of these new security features. Over time, this work improves security for everyone in the ecosystem.

Making investments in the future

As noted above, we are making progress, but our work on software supply chain security continues. We recognize that public and private sector stakeholders need to make significant investments for the future to improve security for everyone. At Google, we are committed to doing our part. This includes several [significant financial commitments](#) outlined above and support to third-party foundations like [OpenSSF](#) that manage open source security priorities and help fix vulnerabilities.

We also started the [OpenSSF Alpha-Omega and SOS projects](#) to improve the security posture of critical

open source projects via directed funding efforts. This includes funding to hire security professionals, conduct security audits for critical projects, and provide assistance for incorporating security tools as part of a secure software development lifecycle. Google has spent [\\$7.5 million in various open source security efforts in the last year](#). We encourage governments and industry partners to support those and similar projects.

We also welcome opportunities to participate in important forums like the [Cyber Safety Review Board](#) and the [White House Open Source Software Security Summit](#), and look forward to working alongside others to continue to protect the global software supply chain ecosystem (for more information on the policy recommendations we support, see Appendix A). It's clear that public and private sector stakeholders learned a great deal from SolarWinds, Log4j, and other software supply chain events and this report provides an in-depth review of shared challenges and potential solutions. Now, we must act on those learnings to improve the security of the entire ecosystem.



Appendix A:

Policy checklist for improving resilience of the software supply chain

For improving software supply chain resilience, policy-makers should consider the following measures:

- Ensure appropriate prioritization of cybersecurity: Given widespread reliance on digital tools and applications for almost every aspect of modern society, cybersecurity efforts must be appropriately prioritized and resourced, and should be made accountable to senior executive officials as a core aspect of an entity's ability to execute its mission.
- Consider security requirements for software procurement: Entities should consider establishing security requirements as a prerequisite for vendor eligibility to provide software solutions. These security requirements should leverage existing international, consensus-based standards to the extent possible to avoid fragmentation across jurisdictions and should include training for procurement officials. These initiatives could also include measures such as requiring SBOMs used in concert with frameworks such as SLSA and improving functional security specifications for SaaS software.
- Map your software supply chain: Organizations should make an effort to understand their key terrain when it comes to critical pieces of software in use on their systems, where that software comes from, and the security practices employed in the development and deployment of the software, whether proprietary or open source.
- Ensure coordinated vulnerability disclosure plans are in place: These should include clear policy for partnering with software vendors to understand new vulnerabilities, compiling and disseminating cybersecurity guidance for mitigating newly discovered vulnerabilities, and implementing disclosure timelines that are transparent and that respect requested embargoes to allow for patch development.
- Identify potential software supply chain risks: Once an organization has a clearer picture of their software landscape and supply chain, they should identify associated risks (whether due to supplier practices, open source software provenance, exposure to known vulnerabilities, or dependencies between applications).
- Pursue industry partnerships: Industry and government have distinct, important roles to play in advancing cybersecurity generally, and software security specifically. Industry partnerships are critical to ensure development of well-informed policy measures (and technology solutions) that enhance resilience to cyber threats, while also promoting (rather than inhibiting) innovation.
- Develop software supply chain risk mitigation plans: As risks are identified, organizations should ensure appropriate risk mitigation plans are in place, including vulnerability management (prioritizing timely patching), and incident response plans. These plans should also be clear and concise, and should be exercised and updated regularly.

Appendix B:

[Know, Prevent, Fix](#) Approach to Help Secure Software Supply Chains

At Google, we've written about key principles ([Know](#), [Prevent](#), [Fix](#)) to help organizations address vulnerabilities in open source software. To support this effort, we're working with the community to develop and drive common goals across the ecosystem to provide every open source developer with effortless access to end-to-end security by default. We've provided more information on our progress to date below and encourage all organizations to support these efforts and leverage the tools and resources to [build a safer open source community](#).



Principles	Goals	Community Progress and Available Tools
Know about the vulnerabilities in your software	<ul style="list-style-type: none"> Capture more precise data about vulnerabilities Establish a standard schema to track vulnerabilities across databases Create tooling for better tracking of dependencies 	<ul style="list-style-type: none"> New vulnerability format adopted by several open source ecosystems (Python, Rust, Go) and vulnerability databases (GitHub's Security Advisories and the Global Security Database) New CVE 5.0 JSON schema improved interoperability and OSV.dev now includes searchable vulnerability database Open Source Insights project launched to help community understand the impact of Log4j and Google published data powering project with public Google Cloud Dataset GUAC project launched to support better comprehension of supply chain metadata
Prevent the addition of new vulnerabilities	<ul style="list-style-type: none"> Understand risks when deciding on a new dependency Improve development processes for critical software 	<ul style="list-style-type: none"> Community launched Security Scorecards to evaluate a project's adherence to security best practices and partners now conduct regular Scorecard scans of over one million projects Sigstore created to help organizations sign, verify and protect software and is now generally available, Kubernetes uses it to sign its releases SLSA framework community added new contributors and introduced improvements OSS-Fuzz service helped open source developers fix 2200 vulnerabilities across 500+ projects in the past year and now includes support for new languages such as Java and Swift Linux Kernel Self-Protection Project overhauled internal kernel APIs to detect and stop buffer overflows
Fix or remove vulnerabilities	<ul style="list-style-type: none"> Understand your options to remove vulnerabilities Enable notifications to help speed repairs Fix the widely used versions of affected software 	<ul style="list-style-type: none"> Alpha-Omega project issued first grant to the OpenJS foundation to support vulnerability remediation Google provided \$1 million to SOS Rewards, and \$300,000 to the Internet Security Research Group to improve memory safety by incorporating Rust into the Linux kernel Google employees contributed hours, effort, and code to tens of thousands of open source repositories, and we created a new Open Source Maintenance Crew to work closely with upstream maintainers on improving the security of critical open source projects

Thank You for Reading

To learn more visit cloud.google.com/security

