THE PRESIDENT'S NATIONAL SECURITY
TELECOMMUNICATIONS ADVISORY COMMITTEE

DRAFT NSTAC REPORT TO THE PRESIDENT

Software Assurance in the Information and Communications Technology and Services Supply Chain

TBD

Contents

Figures

Tables

Executive Summary

Report Focus and Scope

This report focuses on software assurance and the information and communications technology (ICT) and services supply chain. It addresses the results of phase one of the President's National Security Telecommunications Advisory Committee (NSTAC) multi-phase study on "Enhancing Internet Resilience in 2021 and Beyond."[1] The NSTAC study paper that commissioned this work states:

> Recent software supply chain compromises highlight critical risks and large-scale ramifications for industry and Government. The exploitation of software products on sensitive systems—including those performing essential business, national security, or safety functions (e.g., operational technologies, Industrial Internet of Things networks)—can have significant impacts on the United States' national security and emergency preparedness missions.

Underscoring the urgency of addressing these risks, President Biden on May 12, 2021, issued Executive Order (EO) 14028, *Improving the Nation's Cybersecurity*.[2] The EO outlined several new requirements related to software security and addressed additional responsibilities and practices, including new priorities for the Department of Homeland Security (DHS) Cybersecurity and Infrastructure Security Agency (CISA) and for the Department of Commerce's National Institute of Standards and Technology (NIST).

While much of the implementation has yet to be defined, the White House has provided valuable direction and a framework for the Federal Government's cybersecurity efforts. Government-private sector collaboration will be important to extend progress. For example, the existing DHS ICT Supply Chain Risk Management (SCRM) Task Force gathers security expertise from the information technology (IT) and communications sectors, as well as government agencies, thus offering one avenue to address any new sector-specific implementation guidelines for the implementation of EO 14028.

Given the increasing use of and dependence on software in critical infrastructure, this report identifies several areas for urgent action. To address these areas, **the President should establish a task force charged with defining a public-private initiative focusing on key areas of software assurance and the software supply chain. Like the earlier public-private effort on the NIST Cybersecurity Framework (CSF),[3] such an initiative can address fundamental misalignment of incentives, diversity of the assurance approaches, and complexity of the software supply chain. An effort of this nature can translate the urgent need for action into an implementable framework.**

---

[1] President's National Security and Telecommunications Advisory Committee (NSTAC), "NSTAC Report to the President on Communications Resiliency," April 2011, https://www.cisa.gov/publication/2011-nstac-publications

[2] Executive Order (EO) 14028, *Improving the Nation's Cybersecurity*, The White House, May 12, 2021, https://www.whitehouse.gov/briefing-room/presidential-actions/2021/05/12/executive-order-on-improving-the-nations-cybersecurity/

[3] National Institute of Standards and Technology (NIST) "Framework Documents," https://www.nist.gov/cyberframework/framework

The task force should include workstreams to define and help execute the recommendations below requiring public-private partnership as well as software assurance and SCRM expertise.

Key Findings and Recommendations

The subcommittee's findings and recommendations, summarized in tables below, fall into three main areas of focus: software assurance, stakeholders, and external influencing factors.

Software Assurance

Table 1: Software Assurance Findings

Finding	Key Points
No single software security assurance approach works for all situations and environments.	• No single set of software assurance practices can address every situation, due to the diversity of computing environments and development and deployment practices. • Standards and frameworks can guide organizations, but each organization needs to tailor the best solution based on standard components and best practices adapted to the optimal approach for the organization in question. • While there is no single approach for all environments, establishing a high-level baseline for software assurance practices while reducing the complexity of sector-specific regulations is likely to improve the efficiency of adaptations organizations need to make when defining their software assurance approaches.
Best practices in SCRM are not generally tailored to software.	• While the software industry has ready access to standards and best practices covering SCRM and software assurance practices, these standards and practices need better adaptation to software and modern software development and deployment models. • Adoption of SCRM best practices in both public and private sectors has been uneven.
Open source software (OSS) is not inherently less secure than closed source software, but incentives to invest in securing open source are neither effective nor sufficient.	• Open source software, which provides components for virtually all software products, thrives on diversity of contributions and contributor motivations. Not all contributors are motivated to adopt security assurance practices. • Developers and administrators may not have insight into the level of security assurance for OSS modules. • Various promising efforts are underway that may lead to improved trustworthiness and increased confidence for integrators and users of software products that contain open source. The prospects for success and impact of these efforts are still uncertain

Table 2: Software Assurance Recommendations

Recommendations	Key Points
1.1. The Government and industry must collaborate on broader, actionable adoption of well-established, existing SCRM practices adapted to the modern software ecosystem.	a. Fund NIST to work with industry to identify, evaluate, and measure the effectiveness of new security assurance practices. b. Develop and adapt standards and best practices for secure build environments for software. Tier these practices to allow appropriate scaling based on the criticality of the software and the size of the development organization. c. Examine processes used by organizations focusing on cybersecurity and software assurance and approaches used in industry sectors (e.g., telecommunications) to improve organic best practices in software assurance and software supply chain. d. Groups and task forces formulating software assurance requirements should stipulate that the diversity of developer organizations needs to be adequately represented. Reference DHS's ICT SCRM Task Force[4] efforts as a baseline to assess threat mitigation relative to software assurance.
1.2. Direct NIST to convene a public-private effort to improve harmonization among standards in security assurance.	a. Identify gaps, conflicts, overlaps, and obsolescence in software security assurance standards, guidelines, and frameworks. b. Use the interagency process, public-private partnership, and global leadership to support and leverage relevant efforts, such as the NIST CSF and Secure Software Development Framework (SSDF).[5] c. Update the NIST CSF to refer to SSDF practices that address the capabilities gaps identified during the efforts mentioned above.
1.3. The Government should invest in research and development (R&D) for the software assurance field to keep up with advances in computing architectures.	a. Support R&D in software assurance in Government agencies and labs, academic research programs, and industry to address future computing architectures. b. Invest in innovation to automate software assurance tasks, including auditing, testing, collecting requirements, generating secure code, developing threat models, and software SCRM. c. Strengthen emerging approaches in software assurance, such as using artificial intelligence (AI) and evidence-based data-driven metrics.
1.4. Improve security and assurance processes for OSS.	a. Incentivize collaboration between open source developers and organizations focusing on security, such as the Open Source Security Foundation (OpenSSF).[6] b. Task NIST to extend efforts from its work related to EO 14028[7] to identify the top open source packages used for "critical software." c. Task the Federal Government to engage with organizations, allied nations, and Government agencies outside of the U.S. (e.g., the European Union Agency for Cybersecurity [ENISA],[8] the G7, or the United Nations), to create and fund a public-private software assurance program to improve open source security.

[4] The United States (U.S.) Department of Homeland Security publishes information about its Supply Chain Risk Management Task Force and Task Force Publications at https://www.cisa.gov/ict-scrm-task-force

[5] NIST, "Secure Software Development Framework [SSDF]," https://csrc.nist.gov/Projects/ssdf

[6] Open Source Security Foundation (OpenSSF), "OpenSSF, The Linux Foundation Projects," https://openssf.org/

[7] EO 14028, https://www.whitehouse.gov/briefing-room/presidential-actions/2021/05/12/executive-order-on-improving-the-nations-cybersecurity/

[8] European Union Agency for Cybersecurity (ENISA), https://www.enisa.europa.eu

	d. Develop standards to accurately describe software components, in collaboration with organizations such as OpenSSF and international standards bodies. e. Encourage developers to adopt a system of code vetting, such as OpenSSF's Scorecard 2.0.[9]

Stakeholders

Table 3: Stakeholder Findings

Findings	Key Points
Stakeholders in development, procurement, and administration of software have different requirements and needs that are sometimes in tension.	• The software supply chain frameworks and best practices need to align with the needs of developers (who must innovate rapidly to compete), procurement teams (who must evaluate alternatives against schedule, cost, and performance requirements), and administrators (who must manage and update deployed technology).
It is difficult to provide provable evidence of software security assurance practices.	• Efforts are underway to increase software assurance transparency, but it remains challenging for stakeholders to provide useful evidence of adherence to practice.
Guidelines for software supply chain assurance are not evolving fast enough.	• The nature of software development has fundamentally changed over time and will continue to change. Cloud-based software, more frequent releases and updates, increased use of third-party code modules, and the breadth of open source have all impacted how stakeholders develop, purchase, deliver, and manage software. • Assurance frameworks must also evolve. Keeping pace with evolving technology is essential for building viable software assurance requirements.

Table 4: Stakeholder Recommendations

Recommendations	Key Points
2.1. Incentivize engagement among all groups of stakeholders in software assurance programs, at both the domestic and international levels.	a. Engage all software lifecycle stakeholders (developers, administrators, integrators, procurement, and others) in developing adequate assurance programs with clear guidance and associated best practices. b. Make standardization of software assurance an international effort, as companies are challenged to support an array of different requirements from multiple geographies. c. Partner with international standards development organizations and maintain transparent operations in these organizations.

[9] Mertic, John, Open Source Security Foundation, The Linux Foundation Projects, "Open Source Ecosystem Gains New Support for Securing the World's Most Critical and Pervasive Software," July 28, 2021, https://openssf.org/press-release/2021/07/28/open-source-ecosystem-gains-new-support-for-securing-the-worlds-most-critical-and-pervasive-software/

2.2. Incentivize flexible, easy-to-adopt software assurance practices for developers and suppliers.	a. Require procurement teams to prefer products that address an agency's threat models and adhere to security standards. b. Encourage the U.S. Government and industry experts to engage in global standards organizations to ensure standards are global, flexible, and easy to adopt, to support the breadth of software suppliers (from small and medium businesses to large corporations). c. Promote developer use and adoption of comprehensive software assurance practices for operations, hardware, storage, design, coding, and communications security. d. Help document requirements for comprehensive software assurance programs, spanning threat analysis, vulnerability identification and tracking, ongoing penetration testing, build verification, and attestation. e. Continue to promote adoption of approaches such as the NIST CSF and SSDF.
2.3. Reform and update U.S. Government acquisition regulations to drive better SCRM practices, especially for designated "critical software."	a. Require procurement teams to prefer critical software that has been developed and will be maintained according to supply chain risk management frameworks such as NIST Special Publication (SP) 800-161 or SSDF. b. Establish and support pilot programs around software component visibility and supplier evaluation tools.
2.4. Improve software administrator information sharing practices to increase awareness of, and mitigate risks to, software in use.	a. Identify and resolve current legal, procedural, and personnel challenges to timely information sharing of data on adversary activities and vulnerabilities. b. Authorize specific agencies to expand use of warning systems, educational programs, and document best practices and key lessons learned. c. Provide a single point of contact within the U.S. Government (e.g., CISA within DHS) for industry information sharing to avoid overhead and inefficiencies to industry of multiple information sharing channels. d. Support incident response readiness at the federal and state levels by fostering deeper collaboration between federal, state, and national guard cyber resources and local Information and Communications Technology and Services (ICTS) cybersecurity providers.

External Influencing Factors

Table 5: External Influencing Factors Findings

Findings	Key Points
The global range of suppliers makes it challenging for governments to implement "one-size-fits-all" regulatory requirements.	• The economics of the software industry are unique, and components of critical software may come from suppliers in large corporations, small and medium-sized businesses, or even unknown sources contributing to open source repositories. • For critical infrastructure, sector-specific implementation guidance should be considered.

Security assurance practices are not being taught early, consistently, or broadly enough.	• Computer science and software engineering programs at universities lack consistent requirements on security assurance, and new developers frequently lack skills in security assurance practices. • The lack of both depth and consistency requires employers to invest in costly training programs in security assurance and results in decreased harmonization in terms of skills and expertise, which is especially pronounced in small-medium businesses and in new areas of technology. Especially for smaller employers and startups, this burden diverts resources from developing new features, which increases competitive risk. • While formal training in security assurance is expected to be obtained beyond the K–12 level, introducing cyber security education earlier, during K–12, could increase the numbers of future security experts and the security acumen of citizens in general.

Table 6: External Influencing Factors Recommendations

Recommendations	Key Points
3.1. Task the Government to create a task force to define viable incentives to support assurance practices in the extremely diverse software ecosystem.	a. Appoint a public-private representative task force to evaluate and propose economic incentives to improve software assurance and software supply chain security. b. Invest in research on the economic aspects of software development, deployment, and administration. c. Avoid measures that might hinder technology innovation.
3.2. Harmonize requirements for software security assurance among engineering students and in training programs.	a. Appoint a panel of higher education, industry, and open source community leaders and training organizations with the charter to report, within one year, on recommended core security curricula for software engineering and computer science departments. b. Incentivize inclusion of the core security curricula as graduation requirements for students enrolled in undergraduate programs in computer science. c. Work with professional groups and security training organizations to establish postgraduate competency examinations (much like the professional engineer licensing exam for engineering graduates) and to align certifications against new requirements.
3.3 Encourage introduction of security concepts in K–12 education.	a. Encourage states to introduce security education in K–12 curricula.

1. Introduction

1.1. Background and Motivation

The President's National Security Telecommunications Advisory Committee (NSTAC), consisting of industry chief executives from major telecommunications, finance, and aerospace companies, as well as network service and information technology providers, has provided timely and actionable recommendations to the President for more than thirty years to safeguard a reliable, secure, and resilient national communications infrastructure through any event or crisis.

Pursuant to this longstanding mission, the NSTAC formed the Software Assurance in the Commercial Information and Communications Technology and Services (ICTS) Supply Chain Subcommittee in May 2021 to conduct the first part of a four-phase study on "Enhancing Internet Resilience in 2021 and Beyond." The entire 18-month study (from May 2021 to November 2022) will provide actionable insights on how the United States can improve security, develop a more robust digital infrastructure, and provide a more resilient digital experience.

Formation of the Software Assurance in the Commercial ICTS Supply Chain Subcommittee was timely. Recent sophisticated and malicious cyberattacks have disrupted and imposed significant costs on both the private and public sectors. These attacks highlight risks to the U.S. digital economy and threaten the resilience of critical infrastructures in times of crisis.

Both risks and resilience are moving targets, as nationwide and global reliance on digital and digitally dependent infrastructures continues to increase. Smart technology and the Internet of Things (IoT) continue to evolve, making internet accessibility central to more and more activities. This evolution, in turn, extends the attack surface significantly.

High-profile supply chain attacks underscore the depth and breadth of the challenge of software assurance and the importance of a high-integrity, secure software supply chain to critical infrastructure companies and agencies:

- In December 2020, a cybersecurity firm discovered that attackers had infiltrated the code-building environment of a major developer of information technology (IT) management and monitoring tools, implanting malware later named "SUNSPOT." SUNSPOT placed a vulnerability in tool code that customers had downloaded in at least two software updates. The attackers used the vulnerability to gain access to customer IT systems.[10] [11]
- In July 2021, an IT solutions provider inadvertently pushed infected software to its customers, sparking a widespread ransomware attack that impacted roughly 60 managed service providers and around 1,500

[10] MITRE ATT&CK, "SUNSPOT," January 12, 2021, https://attack.mitre.org/software/S0562/

[11] SolarWinds, "SolarWinds Security Advisory," April 6, 2021, https://www.solarwinds.com/sa-overview/securityadvisory#anchor1

businesses, mostly small and medium-size enterprises.[12] This ransomware attack was only the most high-profile attack of a year when, according to the Department of Justice, $350 million in ransom was paid to cyber criminals, a more than 300 percent increase from the previous year.[13]

Such supply chain attacks demonstrate that the threat landscape encompasses the entire software lifecycle: development to implementation to maintenance, including design, coding, and testing, as well as cyber hygiene practices such as updating and patching.

Also of concern, given the NSTAC's focus on the U.S. telecommunications infrastructure, was the May 2021 ransomware attack on a major U.S. oil and gas distribution enterprise.[14] This attack was a striking reminder of the importance of internal governance and situational awareness of information and communications technology (ICT) systems within critical infrastructure companies. The attackers exploited a legacy virtual private network (VPN) profile that was still operational and accessible without multifactor authentication (MFA) – a violation of foundational best practices. Additionally, while the attack occurred within the energy provider's back-office IT network, it was uncertain whether the infiltration also impacted the operational networks that governed its pipelines. This uncertainty prompted the shutdown of the pipelines, triggering economic and societal disruptions across the U.S. economy.

These attacks cut across IT systems in different sectors of the U.S. economy. Their scope is prompting a whole-of-government and whole-of-economy reassessment of effective, resilient cybersecurity best practices in critical areas such as operational security, software assurance, and supply chain integrity. Following the resolution of one of these major attacks, President Biden issued Executive Order (EO) 14028, *Improving the Nation's Cybersecurity*,[15] which outlined several initiatives leveraging the Federal Government's ICT buying power to improve the overall security and integrity of the software supply chain. In addition, after conducting hearings, Congress is considering possible legislative responses, including incident reporting, new investments in ICT modernization at all levels of Government, and cyber workforce development. This level of Government and policymaker engagement is not exclusive to the United States; similar government examinations of supply chains are underway in other countries.

The work of the Software Assurance in the Commercial ICTS Supply Chain Subcommittee is a critical complement to the work in Congress and the public and private sectors in response to EO 14028.[16] This report focuses on

---

[12] Tung, Liam, "Kaseya Ransomware Attack: 1,500 Companies Affected, Company Confirms," July 6, 2021, https://www.zdnet.com/article/kaseya-ransomware-attack-1500-companies-affected-company-confirms/

[13] Lynch, Sarah, "U.S. Launches Online Hub to Help Ransomware Victims," July 15, 2021, https://www.reuters.com/world/us/us-launches-online-hub-help-ransomware-victims-2021-07-15/

[14] Sanger, David et al, "Cyberattack Forces a Shutdown of a Top U.S. Pipeline," May 8, 2021, https://www.nytimes.com/2021/05/08/us/politics/cyberattack-colonial-pipeline.html

[15] EO 14028, *Improving the Nation's Cybersecurity*, The White House, May 12, 2021, https://www.whitehouse.gov/briefing-room/presidential-actions/2021/05/12/executive-order-on-improving-the-nations-cybersecurity/

[16] *Ibid*

developing pragmatic, actionable, evidence-based recommendations for greater software integrity, broad improvements to software security, and an overall safer computing environment.

1.2. Process and Involved Stakeholders

Subcommittee members worked collaboratively to study current and emerging solutions that could mitigate known and emergent security risks. Based on that examination, the Subcommittee considered and developed findings and strategic recommendations to promote public-private coordination and safeguard mission-critical communications systems to the President.

In examining current and emerging solutions, the subcommittee held numerous briefings with subject matter experts (SME) including experts from:

- Several companies, including companies led by NSTAC members;
- The open source community;
- Leading academic institutions;
- Nonprofit organizations and think tanks; and
- Key U.S. Government agencies, notably the Department of Homeland Security (DHS), the Cybersecurity and Infrastructure Security Agency (CISA), and the National Institute of Standards and Technology (NIST).

1.3. Report Structure

The report highlights the current software supply chain ecosystem; identifies existing stakeholders; outlines several external forces with significant impact on software assurance and its supply chain; and presents findings and recommendations. Appendices include relevant supporting materials such as a table of threat types and lessons learned from assurance programs, as well as participants, acronyms, definitions, and a bibliography. The key areas examined include:

- Overview of Software Assurance and ICT Supply Chains. This section discusses the software development lifecycle, current approaches to software assurance, current technology approaches such as automation and threat modeling, and a listing of key issues.
- Stakeholders. The dynamism and complexity of software development, distribution, use, and maintenance supply chains are reflected in the breadth of the stakeholder community. This section notes typical roles of major stakeholders, the differences between developers (e.g., of open source or proprietary models), buyers, and administrators of software.
- External Influencing Factors. The resilience and integrity of the software supply chain is both positively and negatively influenced by several forces outside the perceived perimeters of the supply chain itself, including the computing ecosystem, relevant economic factors, cybersecurity education and training, regulatory systems, and standardization.

2. Overview of Software Assurance and Information and Communications Supply Chains

2.1. Software Development Lifecyle

The secure software development lifecycle (SSDL) is a multi-phased process to create secure software products. While no single standardized process exists for developing secure software, all software development includes similar stages. Figure 1 shows the traditional Microsoft secure software development process; in general, phases include training, gathering requirements, designing, implementing, verifying, releasing to customers, and responding to continued maintenance and end-of-life decommissioning and disposal. Each phase of this basic lifecycle includes security activities to reduce vulnerabilities in the software product.

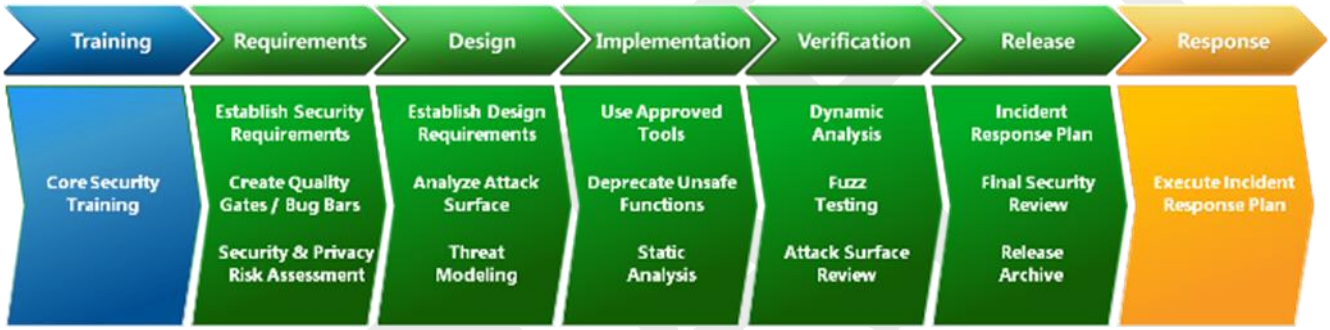

Figure 1. Secure Software Development Process Model at Microsoft[17]

2.1.1. Changes to the Basic Lifecycle

Many elements affect this basic lifecycle. New use cases, software enhancements, and deployment strategies modify the product. Lifecycle stages may be re-sequenced to accommodate new producer and customer needs or expedite deployment. Additionally, some software supply chains include third-party components, which developers download and use in multiple ways, such as:

- Downloading source code so that they can maintain it themselves if needed;
- Forking the code base for their own purposes;
- "Compiling from source" as a mitigation against some supply chain attacks; and
- Downloading pre-compiled binaries they can integrate directly into their software.

Some third-party component suppliers include analyses of their code to help downstream integrators and users determine fitness for purpose and licensing.

[17] Microsoft, "The Traditional Microsoft Product Development Process, The Security Development Lifecycle," May 22, 2012, https://docs.microsoft.com/en-us/previous-versions/windows/desktop/cc307406(v=msdn.10)#the-security-development-lifecycle-sdl

2.1.2. Cloud Services and Delivery Models

The largest change in software delivery and usage is an increasing move from on-premises solutions to cloud services. Cloud services have several delivery models relevant for software assurance:

- The cloud service provider (CSP) manages all elements of the service for their customers (e.g., a business application such as human capital management).
- Customers install and manage their elements of the cloud, while the CSP manages the underlying portions of the stack (e.g., cases such as Infrastructure as a Service [IaaS] or a home-grown application run on Platform as a Service [PaaS]).
- The service delivery is a specific installation that the CSP manages at the customer site.

2.1.3. Cloud Service Security Advantages

A cloud service model offers many overall advantages, including:

- **Cost efficiency**. The ability to complete core maintenance and security management (e.g., apply patches or harden instances) at scale offers a speed and thoroughness to processes that would be expensive and fragmented for customers, particularly small and medium businesses, to do themselves. Given the scarcity of cybersecurity talent, having a service provider manage and secure the service makes economic sense.
- **Speed of updates**. The pace of development activity in cloud services – daily or even hourly code changes – provides distinct customer benefits, including the ability to address security vulnerabilities more rapidly.
- **Automated cost-benefit measurement**. As cloud services automate many development processes, they can also automate and measure "technical debt" (i.e., the cost of a full but costly fix, versus an easier but limited fix), giving service providers a more real-time view into their risk posture.

2.2. Security in Software Development

A secure software development lifecycle requires building security practices and requirements into the software development process. To address the security aspects of the development lifecycle, every role needs to be trained in security, from product managers to architects and validators. These roles do not need to be security experts, but they do require a basic understanding of security practices. Security training also helps foster a "cultural norm of security."

Table 7 identifies critical security activities that should occur at specific times in the SSDL.

Table 7: Critical Security Activities by SSDL Phase

In This Phase	Teams Should:
Requirements Phase	• Gather security as well as functional requirements. • Include security protocols for interacting with other systems. • Make decisions on the product lifetime, requirements for updating software in the field, etc. • Develop preliminary threat models to inform security requirements and scope necessary rigor.
Design Phase	• Define a detailed description of the security features: for example, the Transport Layer Security (TLS) implementation, the versions of TLS supported, and related transport protocols. • Include third-party software libraries for the project to use. • Complete the threat model to define the critical assets in the system, the adversaries to protect against, and how these assets are protected to help select and place security controls in the project. Without that definition, it is exceedingly difficult to determine vulnerabilities. • Obtain a peer review of the threat model and design by security experts to validate the security goals of the product.
Implementation Phase	• Confirm that the tasks being performed are using the best technologies. • Verify usage of the appropriate programming languages to tackle the problems associated with each task. • Understand the security deficiencies of the languages used. • Perform basic quality practices, such as code review of changes and scanning tools. • Use an integrated automated test framework for unit, integration, functional, performance, and security testing. • Confirm the security and applicability of third-party libraries.
Verification Phase	• Conduct positive path testing to confirm features are implemented as defined in the requirements document. • Conduct "negative testing," such as fuzzing, to identify vulnerabilities from unplanned, unintended, or non-designed functions. This process can be partially automated. • Use code coverage tools in testing phases to confirm that quality assurance activities test the software.
Release and Response Phases	• Continue involvement with the software after the software is released or the hosting is established. • Continue to look for new vulnerabilities, including in third-party software. • Deploy updates to address these new vulnerabilities.

2.2.1. Supply Chain Security

In addition, teams should examine the following major security aspects affecting the supply chain:

- **Consider the software supply chain feeding products and the ones being fed upstream**. This consideration should include many of the same techniques for validating in-house-developed software.

such as fuzzing, source code static application security testing (SAST), dynamic application security testing (DAST), and interactive application security testing.

- **Define and apply policies for incorporating third party components**. While industry does an excellent job of running static code analysis and software composition tools to find issues early in the development cycle, the security impacts to the software supply chain are not fully understood. It is unclear how developers choose free and open source software (OSS) upon which to build their code. Further, developers are inconsistent in their security approaches. Note: These issues are not limited to OSS.
- **Leverage existing best practices and frameworks to effectively manage critical information**. Developers should consider adoption of a software identification tagging method (e.g., SPDX,[18] SWID,[19] CycloneDX[20]). Google in July 2021 released the Supply Chain Levels for Software Artifacts (SLSA) framework[21] for "ensuring the integrity of software artifacts throughout the software supply chain." Such best practices and frameworks, when applied across companies and their suppliers, make it easier to identify, maintain, and track the various components of a single software solution.
- **Harden the Continuous Integration/Continuous Deployment (CI/CD) pipeline.** Control the access and security of the pipeline so that only authorized administrators can make build changes to prevent any covert, malicious additions. Employ a defense in depth strategy to build security into the pipelines at any location in the software supply chain.

Even with the best controls in place, no tools exist that can flag all possible suspicious insertions, deletions, or replacements. Software is only as secure as the weakest link in the software supply chain.

2.3. Emerging Technology Approaches

Developers can improve software security by taking advantage of the scalability and rapid security updates that cloud-delivered services offer and incorporating SSDL practices. However, these steps are insufficient by themselves. Attackers can also leverage cloud-delivered services to launch large-scale automated attacks at minimal cost. It is only incrementally more costly to deploy a given exploit against all known vulnerable systems than against a single, targeted, vulnerable system. For example, in March 2021 when product vulnerabilities were disclosed for an e-mail server product, zero-day attacks[22, 23] attributed to HAFNIUM were observed against a significant number of vulnerable internet-facing systems within minutes.

---

[18] Software Package Data Exchange, "The Software Package Data Exchange," https://spdx.dev

[19] NIST, "Software Identification Tagging," https://csrc.nist.gov/projects/software-identification-swid

[20] CycloneDX, "Open Web Application Security Project [OWASP] CycloneDX is a Lightweight Software Bill of Materials [SBOM] Standard Designed For Use in Application Security Contexts and Supply Chain Component Analysis," https://cyclonedx.org

[21] Google, "Securing the Software Development Lifecycle with Cloud Build and Supply Chain Levels for Software Artifacts," July 29, 2021, https://cloud.google.com/blog/products/application-development/google-introduces-slsa-framework

[22] Microsoft, "HAFNIUM Targeting Exchange Servers with 0-day Exploits," March 2, 2021, https://www.microsoft.com/security/blog/2021/03/02/hafnium-targeting-exchange-servers

[23] Cybersecurity Infrastructure Security Agency, "Remediating Microsoft Exchange Vulnerabilities," https://us-cert.cisa.gov/remediating-microsoft-exchange-vulnerabilities

Attackers can effectively monetize a wide range of compromised enterprise systems via disruption as much as theft, as the prolific rise in ransomware attacks makes clear.[24] As threats evolve, however, so do the technologies to counter them. Advances in automation, threat modeling, metrics, and asset inventories all play a leading role in software assurance.

- **Automation**. Manual compliance checks for software vulnerabilities are inefficient, error-prone, and not scalable. Automated tools, on the other hand, review a greater number of security metrics using less resource-intensive processes, with more consistent results, and ultimately, at a larger scale. "Big Code"[25] analytics processes, such as those under review by the Defense Advanced Research Projects Agency (DARPA), could result in automated tools that not only evaluate software assurance but also use capabilities such as probabilistic modeling to quantify the degree of confidence in evaluation.

- **Improved and automated threat modeling**. Although threat modeling has traditionally been largely manual, emerging technology capabilities support some level of automation. Integrating threat modeling into the design phase as a foundational element provides the clearest possible understanding of how the software will interact with external entities or processes and the potential attack surfaces that might be exploited. Emerging threat modeling technologies take advantage of the scale and effectiveness of automation to systematically review vulnerabilities and potential mitigations while minimally disrupting the development process.

- **Metrics**. The ability to use AI techniques to process very large data sets in near real time offers new opportunities to develop evidence-based and data-driven metrics related to current and potential vulnerabilities, their impacts, and possible mitigations.

- **Asset inventories**. Automation and threat modeling come together most acutely in attack surface management. Defending against attacks requires a complete, current, and accurate asset inventory to consistently enforce a network's security policies across the entire network environment. However, many security architectures, including those incorporating zero-trust architectures, focus primarily on improving and securing interactions between assets that are already known and managed. Without a complete and accurate asset inventory, security for large, decentralized organizations can have gaps that may be easily and rapidly exploited. Lack of comprehensive asset management is a security architecture flaw.

EO 14028[26] highlights this issue by calling for "accurate and up-to-date data… of software code or components, and controls on internal and third-party software components, tools, and services present in software development processes," and "audits and enforcement of these controls on a recurring basis." Security architectures therefore must incorporate emerging technology approaches to accurately identify, monitor for, and

[24] Kraning, Matt, Palo Alto Networks, "Internet Operations." Briefing to the SA Subcommittee. Arlington, VA, July 15, 2021

[25] Goldstein, Phil, FedTech, "Defense Advanced Research Projects Agency Explores How to Automate Software Assurance Assessments," July 8, 2019, https://fedtechmagazine.com/article/2019/07/darpa-explores-how-automate-software-assurance-assessments

[26] EO 14028, https://www.whitehouse.gov/briefing-room/presidential-actions/2021/05/12/executive-order-on-improving-the-nations-cybersecurity/

remediate vulnerabilities on all internet-facing assets and uniformly incorporate appropriate software assurance and secure software development lifecycle practices. These technology capabilities should:

- Identify strategies and methods to account for incomplete visibility when implementing next generation security architectures.
- Confirm that organizations understand the operation of their IT systems/networks and those of their suppliers to help create effective security controls and uniformly incorporate software assurance practices.
- Strive to maintain an accurate, real-time asset inventory to defend against internet-facing attacks.
- Move policy regimes toward continuous auditing to enhance security controls.

2.4. Key Security Concerns in the Software Industry

A number of key issues in the software industry make a singular methodology for software assurance extremely difficult, if not impossible. Some issues result from hostile actions of attackers, and others from the broad application of computing devices and diversity of software. Understanding these key issues is crucial to establish strategies for trusted and secure software development across the supply chain. This section covers several of these issues and discusses their challenges to a traditional SSDL process.

2.4.1. Threats

Today's software development environments face an enormous threat landscape that covers the entire software product lifecycle. Breaches can have serious consequences, including loss of intellectual property, reputation damage, introduction of new vulnerabilities, and financial losses, just to name a few. Further, this landscape is not static, but constantly evolving with both new attack methods and attacker capabilities.

Appendix A of this document identifies a collection of known attack methods that might be leveraged against organizations developing software products. The table is not exhaustive of all possible or realized threats, but it is a solid collection of threats that any software development organization should consider.[27]

The threat landscape is expected to continue expanding. Adoption of IoT devices and the use of AI are expanding, and increasingly sophisticated threat actors are both adopting and exploiting these technological advances. Mitigation strategies for addressing this shifting landscape require an intersection between information technology and operational technology. Securing the initial development and trying to prevent network intrusions are no longer sufficient. Continued security in this threat landscape requires active monitoring of both hardware and software.

[27] Areno, Matthew and Martin, Antonio, "Supply Chain Threats-Software White Paper," *Intel*, July 2021, https://www.intel.com/content/www/us/en/security/supply-chain-threat-whitepaper.html

2.4.2. Open Source Software Usage

NIST defines OSS in Suborder 6106.01[28] as "Software that can be accessed, used, modified, and shared by anyone. OSS is often distributed under licenses that comply with the definition of 'open source' provided by the Open Source Initiative and/or that meet the definition of 'Free Software' provided by the Free Software Foundation." Engineers all over the world develop OSS with no barriers to the free flow of innovation.

OSS pervades the ICT environment. One study[29] showed that virtually all commercial software contains some open source components, and some 70 percent of these applications consist of open source code. Programmers like the ability to "crowd-source" code that performs a function that their application needs. Technology companies have incentives to contribute to open source code, as it often advances the adoption of their own products.

While OSS provides benefits such as accelerating innovation, reducing development timelines, and reducing software complexity and bugs, its use may introduce security risks such as those listed in Table 8,[30] either unintentionally or maliciously.

Table 8: Factors Contributing to Need for OSS Security Assurance

Factor	Details
Attack surface	• Many commercial codebases also contain a large number of open source components, a subset of which contain high-risk vulnerabilities of which developers may be unaware
Non-secure development	• Contributors to open source software are not obligated to spend an adequate (or any) amount of time on security
Vulnerabilities	• "Using Components with Known Vulnerabilities" made the Open Web Application Security Project (OWASP) Top 10 Vulnerabilities for 2020 • Vulnerabilities are publicly disclosed in knowledge bases, such as the National Vulnerability Database, which can be used as a resource to develop exploits
Propagation	• Open source projects are usually built on other open source projects and libraries, leading to trees of dependencies that make patching or tracking difficult
Persistence	• Vulnerabilities may persist unmitigated or undetected for years before being identified

[28] NIST, "Open Source Code," December 6, 2018, https://www.nist.gov/document/finals610601ver1pdf

[29] Synopsys, "Synopsys Study Shows Uptick in Vulnerable, Outdated, and Abandoned Open Source Components in Commercial Software," April 13, 2021, https://news.synopsys.com/2021-04-13-Synopsys-Study-Shows-Uptick-in-Vulnerable-Outdated-and-Abandoned-Open-Source-Components-in-Commercial-Software

[30] Poretsky, Scott, et al, Ericsson, "Open Source Software Security in an Information Communications Technology Context – Benefits, Risks, and Safeguards," January 14, 2021. https://www.ericsson.com/en/blog/2021/1/open-source-security-software

However, developers who create and integrate OSS, as well as the users of open source code, have vastly different perspectives (Figure 2 below). Developers who rely on open source components and end-users who deploy software products containing open source components must not take security assurance for granted.

Community development gives OSS the potential to be of higher quality, functionality, and security, but only if its developers are aware of the need for, and devote the time and resources to, secure development. Open source developers may have a wide range of motivations, from the opportunity to develop technical mastery to the opportunity to gain community recognition for their work. Downstream users and integrators of open source cannot assume that robust security assurance practices were used in development.

As a result, OSS has the risk of undiscovered security vulnerabilities. A famous example of this risk is the "Heartbleed"[31] bug in the Open Secure Sockets Layer (OpenSSL) cryptography library, discovered in 2014. At the time, the Open SSL project was seriously understaffed yet was used pervasively throughout the industry. One study[32] suggested that OSS developers spend an average 2.27 percent of their contribution time responding to security issues. One research project[33] found that up to 20 percent of the packages in a popular Linux distribution might be "underproduced," indicating insufficient effort invested in addressing issues.

Common developer practices magnify the risks associated with underproduced code. For example, once a developer identifies OSS to solve a programming need, they download it from a code repository and integrate it into their product. If integrated code is not inspected and subsequently updated, security fixes are not incorporated. Figure 2 also shows this "tree of dependencies" as OSS is reused from one project to another. According to another study,[34] up to 75 percent of code bases that contain OSS have known security vulnerabilities, and this percentage is increasing.

[31] CISA, "Alert (TA14-098A) OpenSSL 'Heartbleed' Vulnerability (CVE-2014-016)," revised October 2016, https://us-cert.cisa.gov/ncas/alerts/TA14-098A

[32] Ham, Haylee, Lifschitz-Assaf, Hila, Nagle, Frank, Wheeler, David A., The Linux Foundation and The Laboratory for Innovation Science at Harvard, "Report on the 2020 Free/Open Source Software Contributor Survey," December 8, 2020, https://www.linuxfoundation.org/wp-content/uploads/2020FOSSContributorSurveyReport_121020.pdf

[33] Champion, Kaylea and Mako Hill, Benjamin, "Underproduction: An Approach for Measuring Risk in Open Source Software," February 27, 2021, https://arxiv.org/abs/2103.00352

[34] Security Magazine, "Synopsys Study Shows 91% of Commercial Applications Contain Outdated or Abandoned Open Source Components," May 12, 2020, https://www.securitymagazine.com/articles/92368-synopsys-study-shows-91-of-commercial-applications-contain-outdated-or-abandoned-open source-components

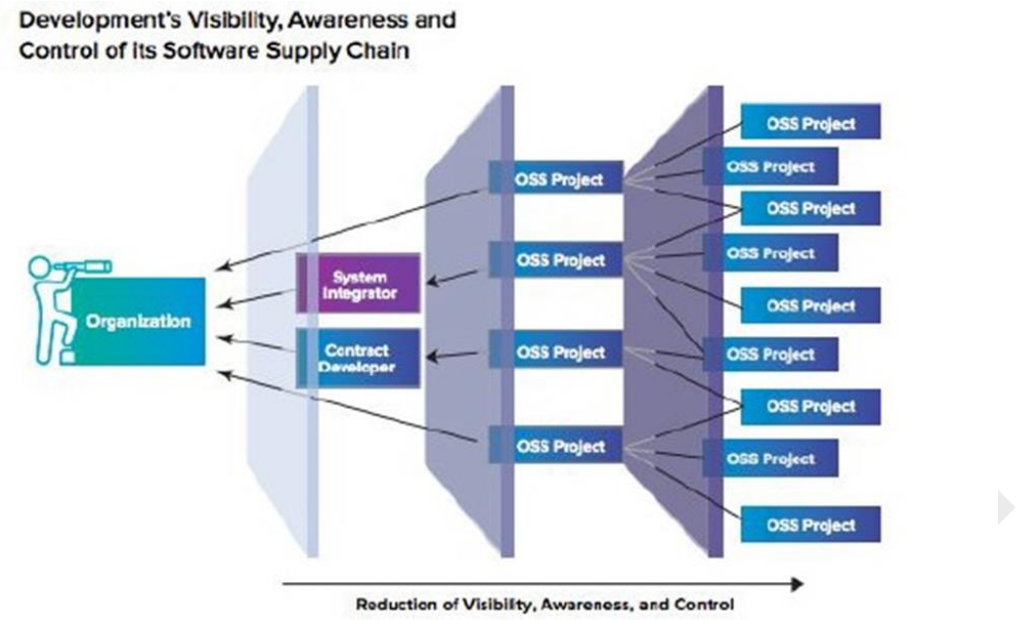

Figure 2: Enablers of Technology Convergence[35]

In short, software vendors need to implement secure software development best practices and cannot rely exclusively upon the open source community to build secure software. Like any code development process, using open source software requires due diligence which organizations can implement by applying industry best practices for supply chain management, secure software development, and secure software maintenance.

When developers update the open source code in their project without applying adequate security assurance practices, they increase the risk of supply chain attacks. In such a scenario, the public repository can be subverted by an attacker and malicious code substituted. The Github 2020 *State of the Octo-Verse* report[36] concluded that a vulnerability on average goes undetected for more than four years before being disclosed; these vulnerabilities live and propagate through reuse of code.

2.4.3. Distributed Development

Multi-Party Development

The nature of software development has fundamentally changed over the last few years. Originally a supplier of applications built all (or virtually all) code from scratch. This practice has been eclipsed by the use of multiple third-party components as "building blocks" for larger, complex applications, including cloud services. By using "pre-fab" components, suppliers can use scarce development resources on innovation instead of "foundational code," as well as shortening their time to market. Also, highly specialized types of code (such as cryptographic

[35] Weeks, Derek, "Introducing Our 2020 State of the Software Supply Chain Report," August 12, 2021, https://blog.sonatype.com/2020-state-of-the-software-supply-chain-report

[36] Github, "The 2020 State of the Octo-Verse," 2020, https://octoverse.github.com/

libraries) can be built once and used many times, potentially affecting security across the spectrum of products and services that use the code.

Secure Integration of Different Software Packages/Components

Incorporating many disparate components into software involves multiple challenges. Some of these challenges existed prior to the expansion of third-party code usage. Securing source code repositories has historically been a high priority for suppliers, as it involves protection of their core intellectual property. And, as SUNSPOT has demonstrated, adequately protecting code repositories, and securing build environments remain essential tasks.

A less obvious challenge is confirming that third-party components can continue to be integrated into supported software. For example, if the originators no longer maintain a third-party component, but a supplier's code base still uses it, the supplier needs both the source code and the capability (and intent) to maintain the code.

While compiling code from source may mitigate some threats (e.g., the threat of a malicious or corrupted compiled component being inserted into the build environment), this approach may impose considerable ongoing overhead costs. When open source components are significantly re-architected from version to version, their revision imposes a code rewrite on those companies to re-integrate the newer version. In some cases, development organizations may not wish to expend the resources to rewrite their code to incorporate a rearchitected version of a third-party component. The security implication is that older versions with extant, known common vulnerabilities and exposures (CVEs) may remain in the code base.

Differences in Security Requirements or Posture

Multiple challenges are involved in using "pre-fab" (typically open source) components to build software applications. The prevalent one is the "tragedy of the commons." Many vendors value the ability to use third-party components (which frees up scarce resources for innovation). However, few vendors improve the security of the open source components they use – not only to avoid the additional cost of improvements, but also because those improvements may benefit their competitors as well.

An additional challenge is that software is neither designed nor necessarily suitable for all threat environments. Without reference to specific threat models, developers have no way to know what threats the incorporated components were designed to withstand. Incorporating a component into a larger body of code does not remediate this problem. Commercial software was not designed for all threat environments, and no amount of retrofitting or code wrapping can change that.

Hardly any suppliers now build all their code from scratch, even though "insourced" development offers more consistent SSDL practices across development teams, including requirements to involve threat and static analysis as part of that SSDL. Otherwise, it is difficult to know what security assumptions or development efforts were used in the incorporated code of multi-party-composed software. Emerging interest in developer intent semantics is a research area that can help increase transparency for such software in service of facilitating greater security assurance.

Software Inventory

Growth in the use of third-party code has eclipsed the ability of those incorporating it to have an accurate inventory of what they are using. Suppliers need to know where their code uses a specific vulnerable third-party library and the status of remediating those instances.

Obtaining an accurate, much less complete, software inventory is difficult. It is not a simple "list of ingredients," because third-party code may include components within other components (e.g., fourth- and fifth-party inclusions in the third-party code). In some cases, developers download a third-party component that may include hundreds of incorporated components, though they intend to integrate only a single component.

An especially complicated problem is obtaining an inventory of all components used in a cloud service. Even defining such an inventory is difficult: should a "cloud service software inventory" include merely the code elements in the "direct" cloud service application (e.g., a human capital management application)? Or should it include the components in all the auxiliary elements of the cloud (e.g., load balancers, firewalls, routers, etc.)? Another challenge is the frequency of changes in cloud services, where code may change multiple times a day. Any inventory, including a published software bill of materials (SBOM), becomes stale almost as soon as it is completed, even with automated SBOM generation.

In many cases, cloud-based transactions involve multiple entities. For example, a transaction submitted to a cloud server may traverse many services prior to completion. Even simple transactions, such as transferring funds from an account in one bank to another bank, may include notices to multiple government and consumer credit agencies. With respect to an inventory, should it include all elements of all services the transaction affects, or a subset? Each application involved in such transactions may be updated independently of the other applications, to the extent that a similar and subsequent transaction a few minutes later might involve newly updated code.

Lastly, an inventory, particularly in generating SBOMs, is not meaningful without standard nomenclature. If supplier and component names are not standardized, it is hard to identify where the code uses a vulnerable third-party library and the status of remediating those vulnerable versions. The criticality of addressing these considerations reinforces the need for standardization to maintain the benefits of an SBOM.

2.4.4. Deployment Diversity

The deployment of software products today varies significantly from one organization to another, often based on the environment of their end destination. How software is deployed and installed in traditional client systems is very different than the processes for servers or IoT systems. Vendors may need to support various deployment strategies, and their strategies may differ from that of their contractors and suppliers. This section explores some of the diversity in software deployment.

Variation in Deployment

Deployment of software products has evolved over the last several decades. In the early days, software was generated using a traditional waterfall sequence (e.g., code, build, test, integrate, final test, and release). Today, more advanced development strategies affect the deployment. Rather than generating a single binary image of a product upon completion, companies are moving to a model of continuous build, delivery, and deployment. This has fundamentally changed how software is ultimately deployed to the end user.

Some products, especially for non-server and non-cloud deployments, continue to use an iterative build process. This typically results in major versions released on a standardized schedule (often annually) with minor versions released either on a shorter standardized schedule or as needed to address functional or security issues. Deployment in this scenario is usually straightforward with versions being available for download manually from a pre-defined website or downloaded automatically by a process performing period checks.

Continuous deployment models result in a cadence of new versions available almost instantly after any modification to the original product. Each new version is subsequently pushed down to the device, where the software is updated on the fly and the new functionality is immediately available to the end users.

These variations in deployment mean that potential attacks, attack points, attack methodologies, remediations, and mitigations vary based on their deployment type. This variance makes it extremely difficult to generate a single strategy to protect the software and its users from attacks.

Lack of Standardization

As stated earlier, no standard method for software development exists, although excellent guidelines and standards providing best practices and guidance for specific areas have been developed. Examples of these standards, guidelines, and frameworks are listed in Section 2.5. Often these guidelines describe only *what* to do and do not mandate the specifics of *how*.

Consider the SSDL process used by the owner of a software product and the SSDL process of any contractors or third-party providers. Currently, no method of standardizing these SSDL processes between the organizations is available. Instead, the owner may merely require that its contractor or providers have an SSDL process and adhere to it during their development, or the owner may forward their SSDL process and require the contractors or providers to adhere to it rather than their own. This latter scenario provides better consistency for the owner, but it can become overly burdensome, if not impossible, for a contractor offering their product to multiple parties with different SSDL processes.

This lack of standardization in SSDL, coupled with different deployment strategies, results in significant challenges in responding to issues discovered in software. Without standardization, it becomes extremely difficult to determine details related to associated vulnerabilities, such as what aspects of the software development process failed to catch the issue and which party was ultimately responsible for failing to detect it.

Conflicting Incentives Across the Software Ecosystem

Software is rarely developed from scratch and depends on code reuse from a wide variety of organizations and open source efforts. Reusing existing code accelerates the development process, since writing code from scratch is labor intensive.

When writing code, how does a programmer choose existing code to leverage? One popular resource is the Stack Overflow website,[37] where programmers share example code, expertise, and suggested solutions to programming problems. Suggested code or pointers to open source projects can be "upvoted" by others. This crowdsourcing mechanism increases a programmer's confidence that the code is appropriate for reuse.

In other cases, the code might be purchased as a supported product from a software vendor. The code being supported by a company rather than a community carries an assumed level of quality.

The unknowns, however, are the assumptions and conditions at play when the code was written: Was the developer considering all the security implications of their code? Was it designed as sample or research code, without being intended for use in production? Usually, the code stands on its own, challenging the programmer to determine if it can be used directly. Table 9 identifies several considerations that compete or conflict with security that could have affected the original code development, which a subsequent user can only guess at.

Table 9: Examples of Competing or Conflicting Incentives in Code Development

Conflicting Incentive	Details
Schedule or budget vs. security	Code is usually written under pressure of schedule, budget, or both. In most cases, programmers are incentivized to meet a scheduled date for completing their functionality and may assume that there is little risk of a security exploit (or that if an exploit is discovered, they will be long gone from the scene). If code is developed with a constrained budget, funding may not cover a security architect, security validator, or security analyst. The code may have potential security bugs, ranging from simple programmer mistakes to fundamental architectural weaknesses.

[37] Stack Overflow, http://stackoverflow.com

Performance vs. security	Performance, whether measured by throughput or latency, is a key characteristic of a hardware/software system. If a service or application performs poorly, it can be a showstopper. Often product updates are touted for their improved performance. However, performance optimizations often open security exploits. For example, many security exploits stem from library functions or methods where the validity of the input values to those functions has not been confirmed. Confirming that the inputs are allowed takes a few additional instructions, and for a function called millions of times, those extra instructions can reduce performance. Thus, the incentive to produce high-performance source code can come at the expense of providing security. In particular, a library designed to provide maximum performance might intentionally skip input checking to save a few cycles, on the assumption that the caller will do the proper checking. If the user of that library doesn't make those checks, it can lead to security issues.
Usability vs. security	Early software was difficult to use and required technical specialists to operate. Every succeeding generation of software improves the user experience and user convenience. However, better usability may come with security or privacy tradeoffs. For example, social media services may have stored private information such as birth dates or phone numbers. This information is a gold mine for criminals since they can exploit it to validate password changes on other internet services or mount phishing attacks.

Even highly skilled and ethical programmers who are properly incentivized to secure their own code, may find their product exploited due to integration of code from a less well incentivized or less ethical upstream contributor.

Heartbleed was one consequence of such mismatched incentives. The OpenSSL project made it easy and free for programmers to encrypt network connections. Network encryption is a laudable goal, since without it, secrets can be intercepted or altered over the public networks. The incentive for encryption was improved confidentiality. But when a programmer submitted some bug fixes, with the incentive to improve performance, a security defect that allowed secrets to be exfiltrated from the application was inadvertently introduced. Thus, the incentives to safeguard confidentiality and improve performance were misaligned with the overarching, implicit incentive to prevent the code from allowing exploits.

2.5. Software Assurance Best Practices

The increased focus on software assurance among businesses and governments produced a range of best practices and guidance for secure software development and software assurance. Initially, individual companies led the development of best practices in response to software security vulnerabilities and concerns, including the advent of the Microsoft Security Development Lifecycle in the early 2000s.[38]

[38] Microsoft, "About Microsoft Security Development Lifecycle," 2021, https://www.microsoft.com/en-us/securityengineering/sdl/about

Seeking to address increasing concerns about the security of commercial technology products and to bolster market confidence in software security, other industry-led groups launched collaborative efforts to drive stronger secure development practices. For example:

- The Software Assurance Forum for Excellence in Code, or SAFECode, a nonprofit entity comprising multiple software development companies, gives member companies a forum to share information on software assurance best practices and to publish guidance for the general public. Examples of publicly available SAFECode products include guidance on fundamental secure software development practices[39] and recommendations for customers to assess the software development practices of their vendors.[40]

Other stakeholder-driven consortia that provide widely used secure software development practices include:

- The Open Web Application Security Project (OWASP), which publishes:
    - The Software Assurance Maturity Model,[41] a technology- and process-agnostic approach to analyze and improve secure development lifecycles.
    - The Software Component Verification Standard, community-driven effort to establish a framework for identifying activities, controls, and best practices, which can help identify and reduce risk in a software supply chain.[42]
- The Building Security in Maturity Model (BSIMM) publishes the BSIMM Framework,[43] which helps organizations assess the software security practices of different organizations.
- The Business Software Alliance (BSA) developed its Framework for Secure Software[44] to provide an outcomes-focused, standards-based risk management tool to help stakeholders communicate security outcomes associated with specific software products and services.

Public-private partnerships and standards development organizations have also produced guidance on software assurance and secure software development. For example:

[39] SAFECode, "Fundamental Practices for Secure Software Development," March 2018, https://safecode.org/fundamental-practices-secure-software-development/

[40] Gilmore, Shaun, Simpson, Stacy, and Sondhi, Reeny, SAFECode, "Principles for Software Assurance Assessment," 2015, https://safecode.org/publication/SAFECode_Principles_for_Software_Assurance_Assessment.pdf

[41] OWASP, "OWASP Software Assurance Maturity Model," https://owasp.org/www-project-samm/

[42] OWASP, "OWASP Software Component Verification Standard," https://owasp.org/www-project-software-component-verification-standard/

[43] Building Security in Maturity Model Framework, https://www.bsimm.com/framework.html

[44] Business Software Alliance (BSA) Framework for Secure Software, https://www.bsa.org/reports/updated-bsa-framework-for-secure-software

- MITRE has issued a white paper and guidance on Principles for Securing Software Supply Chains.[45]

- The Joint Technical Committee of International Organization for Standardization (ISO) and the International Electrotechnical Commission (IEC) has published its ISO/IEC 27034 standard on application security[46] as part of its 27000 security series, demonstrating interest among international standards bodies in secure software development practices.

Government agencies and policy makers are recognizing the important role of software assurance in the broader cybersecurity ecosystem.

- The European Union Cybersecurity Act,[47] which became law in 2019, established an EU-wide cybersecurity certification regime voluntary for general purpose product, including principles on software security by design and process-based certifications.

- NIST launched a collaborative effort with industry in 2018 to develop a Secure Software Development Framework (SSDF)[48] to help software producers reduce vulnerabilities in released software, mitigate the potential impact of the exploitation of undetected or unaddressed vulnerabilities, and address the root causes of vulnerabilities to prevent further occurrences.

- BSA has updated its Framework for Secure Software[49] to align with the NIST work.

- EO 14028[50] includes a significant section on enhancing federal agency and contractor software assurance practices and tasks NIST with publishing guidelines for enhancing software supply chain security, to become mandatory in federal contracts.

- The Department of Defense (DoD) has already piloted secure software development practices in major defense contracts through its use of the "Software Factory," which outlines development, security, and operations (DevSecOps) best practices for automating activities across the SSDL phases.

Significant best practice resources are available to mature software development organizations. However, the increasing use of languages like Python and JavaScript, as well as recent increasing growth in low-code/no-code

---

[45] Clancy, Charles, Ledgett, Jr., Richard H., Nissen, Christopher A., Sledjeski, Christopher L., MITRE, "Beyond SolarWinds: Principles for Securing Software Supply Chains," March 2021, https://www.mitre.org/publications/technical-papers/beyond-solarwinds-principles-for-securing-software-supply-chains

[46] International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC) 27034-1:2011, *Information Technology – Security Techniques – Application Security – Part 1: Overview and Concepts*, https://www.iso.org/standard/44378.html

[47] European Commission, "The European Union Cybersecurity Act," https://digital-strategy.ec.europa.eu/en/policies/cybersecurity-act

[48] NIST SSDF, https://csrc.nist.gov/Projects/ssdf

[49] BSA Framework for Secure Software, https://www.bsa.org/reports/updated-bsa-framework-for-secure-software

[50] EO 14028, https://www.whitehouse.gov/briefing-room/presidential-actions/2021/05/12/executive-order-on-improving-the-nations-cybersecurity/

development environments, indicate the need for more best practices in secure development platforms to enable continued innovation while also protecting security.[51]

2.6. Verification and Testing

Verification and testing (V&T) are important to build assurance across the supply chain, by:

- Creating evidence of software's trustworthiness and discovering indications to the contrary.
- Measuring and improving software quality (e.g., through an automated test framework) and by doing so, reducing opportunities for downstream exploitation of flaws.
- Deterring those who would create flaws or features with malicious intent.

The most basic purpose of V&T is to establish that software behaves as expected under anticipated conditions. Expectations may be set out as requirements for compatibility, performance, functionality, etc. and communicated in use cases, specifications, standards, etc. Finding and fixing errors is also part of V&T, as errors can cause software to behave in ways that fail to meet requirements.

Software that meets all its requirements and is relatively error-free may still be untrustworthy, and best practices for software assurance should continue after the basic goals of V&T are satisfied. The threat environment for software is virtually impossible to anticipate, as it relies mostly on the deployment scenario and not the software itself. As such, the onus of responsibility is on the end user to understand their scenarios and perform some level of V&T themselves.

2.6.1. Techniques

Best practice techniques for V&T consider all phases of the software lifecycle. This section describes methods and tools used by organizations with a mature software assurance culture. The practices are generally ordered along the development lifecycle from concept and design through maintenance and refresh. Common V&T techniques include design characterization, in-line code verification, static code analysis, dynamic code analysis, fuzzing, and continuous verification, discussed in the following subsections.

Design Characterization

Design characterization activities consider the structure of a software-intensive system at a higher level than the code, often at the system level. For example, a criticality analysis may examine all components of a system and identify the ones most critical to the integrity of the system. When those critical components are identified, steps can be taken to reduce or consolidate them and subject the remaining critical components to additional testing and protection measures.

[51] Ware, Bryan, Next5, "The Future of Information Technology," Briefing to the SA Subcommittee. Arlington, VA, August 5, 2021

A related characterization technique is threat modeling, where a model of a system's architecture is created, a spectrum of possible threats is identified in the context of the model, and the result informs design decisions and test cases.

In-Line Code Verification

Just as word processors include automated spelling and grammar checks, most software development toolsets include utilities to check code as it is written. The tools check for obvious errors but can also be configured to check for elements aligned to an organization's coding policies. Policies may explicitly constrain inputs and outputs to and from a module, require units of code to comply with complexity measures, or prohibit certain functions even if they are part of the coding language. These techniques are very effective, but only during the code-writing process. Once code leaves the developer's hands, other methods are needed.

Static Code Analysis

Static code analysis covers a collection of techniques that analyze a body of code; it includes tools that operate against source code and executable code, since both are important. Tools and techniques are purpose driven. The most common tools find errors after some integration has taken place and errors manifest in interplay among several parts of a system in ways that an individual developer could not anticipate. Other static analysis tools look at code for signs of potentially malicious logic or encumbered intellectual property. Recent research is exploring genetic analysis techniques to find code fragments and design patterns with "interesting" lineages.

Dynamic Code Analysis

Dynamic analysis methods observe actual behavior as the subject software executes. Dynamic application security testing is one of many such approaches, observing web applications to find security vulnerabilities. Inputs to the subject software may be structured and targeted to varying degrees depending on the goals of the testing and analysis. Recent AI techniques show considerable promise in creating and automating increasingly complex and insightful test conditions. DARPA's 2016 Cyber Grand Challenge winner is an excellent example of applying AI to perform dynamic code analysis and deploy automated fixes.

Fuzzing

Technically a subset of dynamic analysis, fuzzing is the process of subjecting target software to a large set of inputs purposefully designed to go beyond the expected range of inputs and typically focusing on out-of-bounds conditions. Fuzzing has been employed for years by "vulnerability researchers," both black-hat and white-hat. Applying it to an organization's own code is resource-intensive but is a best practice for code bases that are reasonably expected to be subject to attackers with the means and motive to do their own fuzzing.

Continuous Verification

Most verification methods discussed so far are limited; once the desired results are obtained or metric threshold achieved, the software is delivered or deployed. But no software is ever perfect, and continuous verification

*t*echniques continue to test software after it is operational. Usually building on dynamic testing tools, these environments can aggressively test software essentially nonstop for early discovery of errors and vulnerabilities.

Application to Emerging DevSecOps Environments

In addition to static and dynamic vulnerability assessments, testing should account for emerging DevSecOps techniques and environments as well. For instance, infrastructure-as-code is increasingly used to streamline development processes by enabling software developers to configure cloud resources at scale, and containers are increasingly used to virtualize operating systems; both processes are vulnerable to misconfigurations or other security vulnerabilities. Testing and verification should take into consideration secure infrastructure-as-code processes to identify, prevent, and remediate security misconfigurations (unauthorized privileges, network exposure, public storage buckets) before deployments in the cloud. Container Vulnerability Analysis (CVA) also needs to be performed to evaluate containers against common misconfiguration and software package vulnerabilities and run pre-deployment analysis of third-party images.

3. Stakeholders

The complete ecosystem that enables development and use of secure software is complex, with many actors and influences, but the primary stakeholders are those with the greatest ability to drive positive outcomes:

- **Software developers/solution providers** (including system integrators) create the code base, including application programming interfaces, and produce complete solutions by combining operating systems, applications, and hardware.
- **IT procurement professionals** manage the functional and environmental requirements for product acquisition, evaluate product offerings, implement the contract and service-level agreements (SLAs), and make cost, capability, and risk-based tradeoff decisions.
- **IT deployment and operations staff** integrate, test, deploy, and maintain software within enterprises.

Each of these stakeholders has unique goals and challenges that suggest the existence of a range of obstacles to more secure and resilient software.

3.1. Software Developers and Solution Providers

Software development occurs in wide-ranging organizations and structures with diverse incentives. The following development organizations and their typical software products include:

Table 10: Software Developer Types and Associated Product Types

Development Organization	Typical Software Products
Large multinational corporations writing and maintaining proprietary code.	**Proprietary software** is developed by individuals within the control of one organization who use a logical and standardized system to develop, fix, and maintain software. While this allows greater control of development processes, a more structured development approach, and easier maintenance, it poses obstacles for interoperability and limits outside visibility into the code base (often because much of the code is core intellectual property [IP] of the developing organization).
International open source organizations coordinating the efforts of thousands of independent individuals and groups contributing to their publicly available source code.	**Open source software** is developed by individuals and groups across the globe who develop, fix, assess, and maintain software. The diffuse and unbounded body of potential developers makes OSS difficult to manage and susceptible to tampering. An increasing number of open source components have been integrated into production applications throughout the software supply chain.
System integrators combining software and hardware, often along with middleware, programming interfaces, and microservices, to tie together complex systems of systems for sale to end users.	**System integrators create and implement systems** by combining and customizing hardware and software packages that meet business needs. Integration work often requires creating or customizing code from a variety of sources and vendors to create an end-to-end solution.

These disparate types of organizations share several common goals:

- Meeting customer/market needs to gain and retain market share.
- Rapidly innovating products to address new markets and capabilities.
- Minimizing costs (e.g., through limiting unnecessary work and rework).

3.1.1. Challenges

Today's operating environment presents multiple security challenges to software developers of all types. For example:

- Customer requirements for security vary and are difficult to define.
- Downstream efforts to improve software security require development teams to establish and maintain evidence of process compliance. These process controls increase resource requirements and take time, particularly where different customers have different requirements.
- Maintaining a code base with security updates that address new weaknesses and vulnerabilities as they are identified requires significant resources. After the sale, such maintenance becomes a cost center.

- In complex systems, the boundaries between open source and proprietary software may not be clear, and available security testing tools struggle to support both proprietary and open source development environments, particularly when co-mingled.

- The proliferation of newer programming languages, such as Go and Rust, further complicates assurance. Security testing tools struggle to test all languages, and the tool providers must choose what to support.

Different-sized development firms have unique challenges; small niche companies may struggle to staff dedicated security teams, while larger firms manage very complex CI/CD functions with hundreds of employees around the world. Regardless, mastering these challenges is critical, not only during initial software development but across the full lifecycle; failure to update reused software once it has addressed vulnerabilities can result in component-dependent risks further down the supply chain.

3.1.2. Potential Government Actions Related to Software Developers

Public-Private Partnerships

Elevating the importance of industry-led international standards bodies and increasing participation by industry experts can help new and updated standards work to improve resilience. Government-industry partnership can help reduce the risk of adversaries using standards bodies or Government forums to lessen U.S. competitiveness or introduce weaknesses. Additionally, industry and Government partnerships, including with allied nations, help keep standards-setting processes transparent and independent.

Government and industry should support research and development (R&D) to address future contingencies and identify technologies to improve efficiency, especially in assurance automation and analysis for threat modeling. The DHS ICT Supply Chain Risk Management (SCRM) Task Force has been a success story, showing that collaboration between IT and Communications sectors and government agencies produces meaningful work products for the whole ICT ecosystem. Continuing this work to include sector-specific implementation guidance of the EO 14028[52] software assurance directives is a logical future assignment for the ICT SCRM Task Force.

Incentives

The following actions or initiatives could help make security considerations a priority for organizations and address some of the inherent challenges to software developers:

- Government procurement efforts should consistently **reward vendors who invest in software assurance**. Given the competitive nature of most Government contracts, selecting the lowest-cost bid without weighting attention to software security can disincentivize companies that invest in secure code

[52] EO 14028, https://www.whitehouse.gov/briefing-room/presidential-actions/2021/05/12/executive-order-on-improving-the-nations-cybersecurity/

development. Procurements that place selection value on investments in secure product development incentivize companies to prioritize security.

- Government procurement must **include security as a foundational requirement** on equal standing with functionality and performance. Until security and secure hygiene are intrinsic aspects of procurements, they will remain an afterthought or lower priority, if not a liability (because of cost).

- The Federal Government should **develop joint standards with industry**, including developers and integrators of IoT and industrial control system devices/systems, to feasibly implement and fairly evaluate security requirements.

- The Federal Government should identify appropriate **investments to establish a foundational culture of secure coding**. Initial efforts should work with universities to integrate secure coding throughout computer science and engineering curricula.

- The Government should **support open source software security efforts where possible**, including supporting open source R&D, auditing, training, and education. These efforts should also include sharing best practices among developers, manufacturers, and distributors.

- Government and industry should **support R&D** to address future contingencies and identify technologies to improve efficiency, especially in the areas of assurance, automation, metrics, and analysis for threat modeling.

3.1.3. Secure Software Development Lifecycle Guidelines and Best Practices

The development community leverages a range of guidelines and best practices. Completed and current efforts include:

- From NIST:
  - The SSDF.[53]
  - Special Publication (SP) 800-193, *Platform Firmware Resiliency Guidelines*.[54]
  - Computer Security Resource Center,[55] which provides important developer resources, including guidance on cryptography approaches.

[53] NIST SSDF, https://csrc.nist.gov/Projects/ssdf

[54] Regenscheid, Andrew, NIST, "NIST Special Publication [SP] 800-193 Platform Firmware Resiliency Guidelines," May 2018, https://csrc.nist.gov/publications/detail/sp/800-193/final

[55] NIST, Computer Security Research Center, https://csrc.nist.gov/

- The Open Source Security Foundation (OpenSSF),[56] which offers best practices for OSS projects, including security-related standards.
- SBOMs (like the model and "minimum elements" explored by the National Telecommunications and Information Administration [NTIA][57]) are helpful to the software supply chain, although details of how SBOMs are best used in the industry are still being developed.
- The BSA Framework for Secure Software.[58]
- The Federal Risk and Authorization Management Program (FedRAMP), which offers tools and requirements for developers of cloud services and cloud-based products.[59]
- The National Information Assurance Partnership (NIAP), which helps developers prepare products for security evaluation and Common Criteria certification.[60]
- ISO/IEC 27034 on application security.[61]
- SAFECode's Fundamental Practices for Secure Software Development, Third Edition.[62]
- The SLSA project for creating provenance data for software modules.[63]

3.2. IT Procurement Professionals

IT procurement professionals are tasked with confirming vendors, suppliers, and solution providers meet their organizations' initial user requirements, both functional and environmental; evaluating and down-selecting product or solution offerings; making cost, capability, and risk-based tradeoff decisions; and implementing the contract and SLAs with the vendor or solution provider.

3.2.1. Challenges

Defining the optimal balance of cost, functionality, and security remains a fundamental challenge in IT procurement for several reasons, including:

---

[56] OpenSSF, https://openssf.org/

[57] National Telecommunications and Information Administration, "The Minimum Elements for a SBOM," July 12, 2021, https://www.ntia.doc.gov/report/2021/minimum-elements-software-bill-materials-sbom

[58] BSA Framework for Secure Software, https://www.bsa.org/reports/updated-bsa-framework-for-secure-software

[59] FedRAMP, "Securing Cloud Services for the Federal Government," https://www.fedramp.gov/

[60] National Information Assurance Partnership (NIAP), "About NIAP," https://www.niap-ccevs.org/

[61] ISO/IEC 27034-1:2011, https://www.iso.org/standard/44378.html

[62] SAFECode, https://safecode.org/fundamental-practices-secure-software-development/

[63] Google, https://cloud.google.com/blog/products/application-development/google-introduces-slsa-framework

- As MITRE described in its 2018 "Deliver Uncompromised" paper,[64] Government procurement has **weak incentives for considering solution security**: "Acquisition today is driven to meet cost, schedule, and performance objectives. Absence of incentives for security contributes to widespread compromised systems. Currently, the misalignment of risk and reward during acquisition results in systemic risks being transferred to the operational and sustainment communities without accountability."

- At a basic level, **lack of visibility into the origin of software components** hinders the capacity of procurement officials to reasonably evaluate the security and associated risk of software and systems. Approaches that increase transparency of specific software components, like SBOMs, have the potential to assist but address only a narrow set of risks. Few organizations have the capacity to fully vet second- and third-tier suppliers, and as described below, objective factors prevent complete evaluation.

- **Counterfeit parts**, often difficult to identify until too late, are a longstanding problem in supply chains, including software supply chains.

- Properly evaluating modern integrated systems and components is a highly technical process, and building teams with both the necessary business and technical acumen is difficult. Software purchasers should have **access to trained/certified internal technical and security SMEs** to assist in secure component acquisition.

- Procurement professionals may not have **training on security concepts and risk determination and management** that would equip them to better assess and evaluate the products or solutions.

- A **lack of common systems to accurately and comprehensively price and account for** risk has several effects:

    - It is difficult to justify additional cost for more secure software without commonly accepted and auditable metrics to quantify risk reduction. Both Government and private procurement professionals lack a consistent means to accurately value investments in secure software, putting vendors who bear the costs associated with a secure development environment at a disadvantage.

    - Similarly, procurement teams lack common systems to evaluate tradeoffs in cost and risk for unique security requirements that provide more control and better management of risk in networks and systems. It costs more to implement unique security requirements; off-the-shelf software without such security enhancement is less expensive.

[64] Nissen, Christopher et al, MITRE, "Deliver Uncompromised," https://www.mitre.org/publications/technical-papers/deliver-uncompromised-a-strategy-for-supply-chain-security

- Procurement teams may not have visibility into the after-purchase costs of security integration and security management of a product or solution, hindering an organization's ability to accurately value and account for the security benefits and residual risks of a given acquisition.

3.2.2. Potential Government Actions Related to Procurement

The procurement process is the point at which tradeoffs around cost, security, and other factors are most explicitly evaluated. Opportunities may be available for policy actions that increase procurement teams' visibility into good security practices and accountability and incentives for good security outcomes.

For example:

- The Government could extend partnerships with industry, similar to the model of the DHS Supply Chain Risk Management Task Force, to **develop and document more specific SCRM guidelines** to better measure, assess, and mitigate downstream risk.

- The Government could **update the Federal Acquisition Regulation (FAR)**[65] to include security requirements, attestation of security practices, and SCRM measures on all federal tenders and contracts. It should implement similar measures for critical infrastructure oversight and policies.

- The Government could **reward efforts to educate "digital citizens"** with foundational cybersecurity awareness in the general public (beyond targeting IT professionals) to inform cyber risk-based purchasing decisions by consumers and IT procurement professionals alike.

3.2.3. Procurement Guidelines and Best Practices related to Security

Procurement teams have access to guidelines and best practices for security, some of which overlap with tools and guidelines useful for developers and suppliers. Examples include the following:

- NIST resources, including the Cyber SCRM Framework, SP 800-161, *Supply Chain Risk Management Practices for Federal Information Systems and Organizations*[66] and Internal Report 8179, *Criticality Analysis Process Model: Prioritizing Systems and Components*.[67]

[65] Acquisition.Gov, "Federal Acquisition Regulation," https://www.acquisition.gov/browse/index/far

[66] Boyens, Jon, et al., NIST, "SP 800-161: Supply Chain Risk Management Practices for Federal Information Systems and Organizations," April 2015, https://csrc.nist.gov/publications/detail/sp/800-161/final

[67] Paulsen, Celia, et al., NIST, "NIST Internal Report [IR] 8179: Criticality Analysis Process Model: Prioritizing Systems and Components," April 2018, https://csrc.nist.gov/publications/detail/nistir/8179/final

- DHS ICT SCRM Task Force resources, including a Vendor SCRM template which builds on other industry standards and leverages existing tools and resources, such as ISO guidelines and NIST[68] and an ICT SCRM Qualified Bidder/Manufacturer Risk List.[69]

- FedRAMP, NIAP, and other resources mentioned as tools for developers may be useful for procurement teams as well.

- The emerging Cybersecurity Maturity Model Certification (CMMC) program, though early in its rollout, may eventually help procurement teams confirm that suppliers have basic security controls for their internal information security operations, which could mitigate some risk of supply chain vulnerability.[70]

3.3. IT Deployment and Operations Staff

IT operations teams deploy and maintain software and are responsible for secure configuration and patching and updating software products throughout a product's overall lifecycle. These teams have several key functions:

- Initially integrate and configure new software and systems into existing environments, which are often marked by great complexity and interaction with other enterprise solutions. For example, operations teams must develop, apply, and update IT policies essential to the trustworthy performance of a software solution.

- Patch and update the code base to mitigate known vulnerabilities while simultaneously managing system and resource availability and integrity for mission-critical business operations.

- Address end-of-life and deprecation of critical systems and balance risk if vendors no longer support updates to critical software; essentially, balance continual software assurance assessment with cybersecurity risk, business objectives, and cost.

3.3.1. Challenges

IT deployment and operations staff face multiple challenges, including the following:

- The need to **balance cybersecurity risk mitigation against business costs** for maintenance and operational downtime. Staff must be constantly vigilant in identifying emerging issues and challenges without compromising business goals in an increasingly complex IT environment.

---

[68] ICT Supply Chain Risk Management (SCRM) Task Force, CISA, "Vendor SCRM Template," April 2021, https://www.cisa.gov/publication/ict-scrm-task-force-vendor-template

[69] ICT SCRM Task Force, CISA, "Mitigating ICT Supply Chain Risks with Qualified Bidder and Manufacturer Lists," April 2021, https://www.cisa.gov/publication/ict-scrm-task-force-qualified-lists-report

[70] Office of the Under Secretary of Defense for Acquisition & Sustainment, "Cybersecurity Maturity Model Certification," https://www.acq.osd.mil/cmmc/

- **Maintaining accurate, comprehensive inventories** of the software, systems, network devices, and other information technology, including subcomponent hardware and software.

- **Lack of vendor-supplied visibility into the components and building blocks** for a given product or solution. Accurately managing the ongoing risk of a software solution in the enterprise is complicated when managing legacy infrastructure, difficult-to-obtain support for inherited components from vendors, and third-party code that may not be backed by certified enterprises.

- A **shortage of staff and budget**. IT teams, especially in the public sector, tend to be understaffed and face difficult priority calls in balancing routine policy management tasks with strategic transformation efforts like the shift to cloud-based solutions.

- **Integrating and modifying software and solutions** as part of deploying into the enterprise, which may introduce new risks and gaps in security support coverage.

3.3.2. Potential Government Actions Related to Deployment and Operations

Public sector IT deployment and operations functions should invest in AI and automation to improve efficiency and efficacy of software assurance functions. Operations teams must factor procurement efforts into operational security requirements (costs and benefits during the deployment and maintenance phases of contracts) in evaluating competitive bids.

3.3.3. Guidelines and Best Practices

When vulnerabilities are identified, operations teams must make difficult decisions about how to prioritize patches and updates, which can put business continuity and uptime at risk. Tools to inform operations teams about the existence of and severity of vulnerabilities include commercially available vulnerability scanning tools, which may be used in conjunction with a supplier's SBOM and with NIST's National Vulnerability Database and Common Vulnerability Scoring System.[71]

3.4. Implications for National Security and Emergency Preparedness Communications

The lack of alignment across stakeholder incentives stems from the fact that the software market is not designed to address national security and public safety threats. As a result of misaligned incentives, different stakeholders place different degrees of emphasis on security assurance. Adversaries may be able to identify vulnerabilities and to mount exploits in gaps inadvertently created by market factors. Alignment requires that stakeholders share common definitions, metrics, and frameworks so that each stakeholder group can evaluate measures to reduce risks.

[71] The Forum of Incident Response and Security Teams, Common Vulnerability Scoring System Special Interest Group, https://www.first.org/cvss/; NIST National Vulnerability Database, https://nvd.nist.gov/vuln-metrics/cvss

Greater transparency in the software market is valuable, but such visibility requires tradeoffs among software developers, IT procurement professionals, and IT deployment and operations staff. For example:

- Identifying common vulnerabilities and exposures and enumerating common weaknesses allows rapid assessment and creates opportunities to mitigate risk but may also highlight areas for bad actors to target their efforts.

- OSS allows community review and enables "crowd-sourced" remediation of vulnerabilities but can also allow attackers to inject new vulnerabilities or exploit zero-day vulnerabilities.

- Developing and deploying software security patches can be resource-intensive for both the development organizations and the operations teams. While patching is important, IT operations teams and developers must evaluate the benefit of patching against development, testing, and deployment costs and potential disruptions to users.

4. External Influencing Factors

Software assurance and the software supply chain are large and complex. Figure 3 shows the many internal and external dependencies that go into a software product beyond the supplier's code, including software and non-software factors. This section discusses some external factors in more detail.

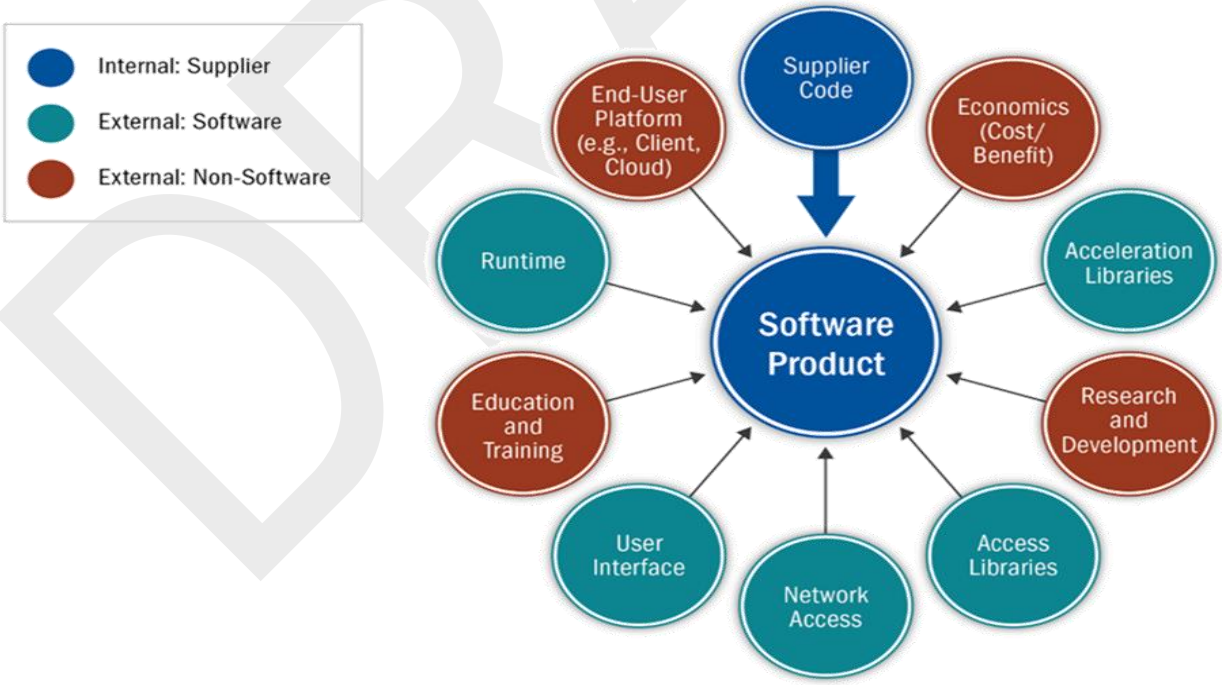

Figure 3: Internal and External Influencing Factors[72]

[72] Hoffman, Robert, Broadcom, "Internal and External Influences," August 2021

4.1. The Computing Ecosystem

The ICT industry is often called an "ecosystem" because it resembles an interconnected web of life. Early computing environments resembled monocultures; a single organization (such as IBM or the BUNCH) created them and exerted a high degree of control on the resulting vertically integrated solution. In contrast, modern solutions are not a single "application." Figure 3 shows clean lines of connection, but the reality is a web of interconnected subsystems, created over time by a large portfolio of unrelated organizations, based on different systems platforms and often written in many computer languages and disciplines.

As a result, a software supplier operates as part of a dynamic build environment, and the supplier's product is a mix of code created by the supplier's programmers plus code obtained from other external sources throughout the ecosystem. This external code might include:

- **Runtimes for so-called interpreted languages** such as Python, PHP, Node.js, and Java. These runtimes execute the basic operations of the language itself, often compiled "just in time" via a JIT interpreter for performance optimization.
- **Acceleration libraries**, which take advantage of low-level hardware capabilities, such as cryptographic functions, graphics functions, or codecs.
- **Access libraries** for functional data sources supplied over the internet, such as geographic mapping functions or public information sources.
- **User interface components**, which reduce the complexity of interacting with pixels and input devices across a broad variety of access methods, such as the web or mobile devices.
- **Network access**, which might encompass physical devices such switches, network gateways and access points, as well as the capability to enable ethernet, Wi-Fi, Bluetooth, 5G, or other public switched data networks.

4.1.1. Software Build Environments

As a discipline, software engineering has evolved to tame and leverage the diversity of the computing ecosystem to produce stable and functional solutions. One common practice is to do a software build, which freezes all the components into a functional whole that can be tested. If the tests prove the build is valid, the software components are digitally signed to prove integrity and are documented in a SBOM to demonstrate provenance.

The build environment puts together reproducible software, which is usually deployed on-premise at a user's data center or device. The software is provided to end users as a bundle, which may include multiple modules and usually includes a cryptographic signature to validate that the software has not been tampered with and was authored by the software vendor. This bundle also may include automated tasks to validate the security of the software. The software is installed by users and, after validation, put into production.

The build environment may also build software to deploy in a software-as-a-service (SaaS) model in the cloud. SaaS applications provide some functionality over the network, and the resulting software is usually not distributed to users for installation.

Other common build environment models currently include:

- **Continuous integration/continuous deployment.** A variant of the above SaaS model, the software is usually installed in a subset of the cloud for immediate feedback and A/B testing in which a functional change is introduced to some percentage of a site's users to compare efficacy.

- **Building software as part of a rapid iterative cycle** (e.g., using an Agile development method). The resulting software may be distributed to users, or it may be used for testing without distribution.

Common to all models are tasks associated with architecting, implementing, and maintaining or optimizing the build process, as well as provisioning and configuring equipment as build servers or virtual machines (including networking and user permissions).

As recent high-profile supply chain attacks have shown, an attacker can modify the build environment to introduce malware into a final product or a product update. Therefore, a build environment must be developed and maintained with the same level of security, integrity, and diligence as the source code and resultant product.

Unfortunately, the task of creating and maintaining the build environment is often relegated to junior engineers or even interns, as a training task. Given recent events, standards organizations, software producers, and threat models need to focus on build environments to drive best practices for their design, implementation, and maintenance. This shift of focus requires significant investment by the ICT industry and, in most cases, requires re-engineering of the process to impose controls that thwart attacks on the build environment or that make the build environment much more resistant to compromise.

4.1.2. Diverse Cloud-Based Architectures

As noted in Section 2, the largest change in software delivery and use is the increasing move to cloud services. A key driver of this shift to the cloud is the growth of several differing delivery models.

In many instances, customers use cloud service provider (CSP) management at the enterprise level (e.g., a centralized SaaS model). In some variations of CSP architecture, the CSP provides service delivery of a specific install at the customer site. And in other variations, customers opt to deploy their own set of software products on top of the CSP's platform or infrastructure.

These different approaches are part of a larger trend of ubiquitous connectivity between people or things or both, aided by the acceleration of wireless and 5G services. This trend influences the SSDL for cloud-based applications and presents distinct challenges and increased risk for software assurance. Cloud-based environments offer malicious actors lower costs and more opportunities to deploy automated attacks at scale. Two examples of this increased risk include:

- **Web applications**. The acceleration of cloud-based services has made web applications a prime target for attackers. Verizon's 2021 *Data Breach Investigations Report*[73] found that 39 percent of all breaches targeted web applications, double the percentage for 2019.

- **Application program interfaces (API)**. The acceleration of 5G and advances in broadband expand the potential for attacks targeting APIs and other points of interconnection.

These developments have motivated increased focus on zero-trust architectures. They also highlight the need for responsibility and accountability models that specifically address physical security (for CSPs) and application security (for software developers).

4.1.3. Innovation

The growth in cloud-based architectures and services is part of a broad innovation push across the computing ecosystem. When innovation impacts the security and integrity of the computing ecosystem, and software assurance in particular, malicious actors and software developers innovate as well by leveraging emerging technologies. Two key examples of this trend include:

- **Automation**. Section 2.3, Emerging Technology Approaches, highlighted the use of large-scale, automated attacks. Automation:

    - Has expanded both the threat and opportunity landscape for attackers at a minimal incremental cost.
    - Is poised to drive major changes in software development generally, and cybersecurity specifically, through big data, ML, and other AI tools.
    - Facilitates new forms of identifying vulnerabilities in design, threat modeling, and remediation.
    - Can improve the effectiveness of build environments, such as auto-remediation of code from external sources, and application security testing, such as automated scans and continuous testing capabilities.
    - Can provide, with AI, an avenue to consider evidence-based data driven metrics in software assurance and software supply chain security.

- **Ransomware.** The rise in ransomware attacks has occurred at the same time ransomware affiliates are using ransomware toolkits under a service model. Most recently, the affiliate allegedly responsible for the Kaseya attack, REvil/Sodinokibi, was reportedly shut down. Shortly afterward cybersecurity researchers discovered evidence of a surge in LockBit ransomware activity, suggesting that ex-REvil attackers had shifted to using LockBit.[74] The notion of ransomware affiliates having access to more than one ransomware toolkit comes at a time when such toolkits are being offered as a service. LockBit, for

[73] Verizon's most recent *Data Breach Investigation Report* is available for download at https://www.verizon.com/business/resources/reports/dbir/

[74] Broadcom, "Affiliates Unlocked: Gangs Switch Between Different Ransomware Families," August 12, 2021, https://symantec-enterprise-blogs.security.com/blogs/threat-intelligence/ransomware-trends-lockbit-sodinokibi

example, launched its "ransomware-as-a-service" model early in 2020. This shift in business models and tactics is likely to drive new capabilities to maintain market viability in this increasingly service-oriented market.

4.2. Economic Factors

Recent independent research[75] has examined ways to allow end-users to undertake cost/benefit modeling of their cybersecurity operations based on existing standards, such as the 2014 NIST Cybersecurity Framework (CSF).[76] Victims of malicious attacks can estimate their financial, operational, and reputational costs, but consistent, transparent modeling is not available to evaluate potential returns on investment, particularly for small and medium firms, to improve the safety and resilience of their systems and the cyber-hygiene of their workforce.

While the NIST framework provides guidelines for individual organizations, it lacks sufficient guidance to conduct cost-benefit analysis to determine an appropriate budget for cybersecurity activities or to determine the most appropriate implementation tier for a given threat model.

EO 14028[77] introduces elements of a cost/benefit calculus to the software-buying decisions of federal agencies, on the theory that, in its role as a software buyer, the Government can be effective in advancing software quality and assurance across the supply chain. Studying the Federal Government "use case" to evaluate the whole economy approach to cost/benefit analyses, standardized through testing and modeling, would benefit private sector organizations that manage critical infrastructures or functions and give small and medium enterprises clarity on potential returns.[78, 79, 80]

4.3. The Cybersecurity Education Problem

Cybersecurity education tends to focus on producing individual cybersecurity experts, but the goal should be to foster a computing ecosystem that avoids many of the security issues in software development and deployment. This broad education outcome requires a drastic restructuring of computer security and related disciplines in colleges and universities, involving sweeping changes such as:

---

[75] Gordon, Lawrence A., Loeb, Martin P., and Zhou, Lei, "Integrating the Cost-Benefit Analysis into the NIST Cybersecurity Framework via the Gordon-Loeb Model," *Journal of Cybersecurity, Volume 6, Issue 1*, March 30, 2020, https://academic.oup.com/cybersecurity/article/6/1/tyaa005/5813544

[76] NIST maintains a website to host the Cybersecurity Framework and related supporting papers and guidelines, here: https://www.nist.gov/cyberframework

[77] EO 14028, https://www.whitehouse.gov/briefing-room/presidential-actions/2021/05/12/executive-order-on-improving-the-nations-cybersecurity/

[78] Bergin, Thomas J., "Mainframe Computers," American University, 2012, http://ds-wordpress.haverford.edu/bitbybit/wp-content/uploads/2012/07/Lecture-9-Mainframe-Computing.pdf

[79] Github, "Supply-Chain Levels for Software Artifacts Framework," https://github.com/slsa-framework/slsa#readme

[80] Github, "Security Levels," https://github.com/slsa-framework/slsa/blob/main/docs/levels.md

- Accreditation bodies changing the core educational requirements in computer science and related disciplines to appropriately incorporate security into all classes as a core ethos.

- A competency examination post-graduation (much like the professional engineers' licensing exam for engineering graduates).

4.3.1 The Current State of Cybersecurity Education

The lack of a solid foundation in secure engineering in computer science (CS), software engineering, and related programs is one of the underlying causes of "the software assurance problem," even though many in the software industry have long expressed the concern. The curricula of many universities have not effectively integrated secure development practices, the rationale for security to be part of design, development, and delivery, or the security ethos that it is every developer's job to understand and embed security throughout system design, development, deployment, and maintenance.

Currently, colleges and universities largely focus on the technical aspects of security, such as cryptography. And while technical security mechanisms are very important to address specific threats, an understanding of secure engineering (in particular, secure development practices) is foundational and applicable to all CS and software engineering graduates. Some describe this approach to instruction as specialized "training," but it is better viewed as foundational knowledge, directly comparable to the knowledge expected of an engineer schooled in structural engineering when designing a building for a relevant "threat environment" (such as the presence of seismic faults).

As a result of this foundational gap, virtually every organization with software development teams must train CS and software engineering graduates on fundamental secure development practices and concepts, such as the importance of correctly validating all input. Companies are inculcating their own version of security into developers, many of whom think "security" is someone else's job (e.g., performed during quality assurance, or by penetration testers). This situation results in multiple approaches to security which are not standardized and may be inconsistent. A strong, standard security foundation shared by all CS and software engineering graduates would increase the general level of expertise and harmonize best practices in the broader computing ecosystem.

4.3.2 Restructuring Computer Science Programs

Fixing this foundational problem is a critical step that will also increase not just the number, but the quality, of experts to improve the state of cybersecurity. To start, CS, software engineering, and related programs must be restructured so that sequences of classes build on foundational security principles and incorporate the security of the solution in assignments. Civil engineers take fundamental classes in structures, with subsequent higher-level classes building on that foundation. Similarly, security is foundational for computer science, so that before something is envisioned, designed, or coded, the developer understands the threats the resulting software needs to meet, architects it to mitigate those threats, and keeps the code free (or largely free) of coding errors that can compromise security.

To scale and enforce this fundamental change in curricula across the U.S. college and university system, relevant accreditation bodies for computer science and those for related degrees (e.g., the Association of Computing Machinery) should examine and modify the curriculum requirements before the degree programs can be certified. Accreditation bodies enforcing such changes would help broadly integrate required cultural norms as well as specific knowledge.

The Federal Government is in an excellent position to support this change in university computer science programs through grants and scholarships. It is important to emphasize that the core area needing change is Computer Science and related curricula, and that the problem of improving software security cannot be solved by creating separate cybersecurity programs of study.

After the Soviets launched Sputnik, the U.S. educational system saw a revolution emphasizing science and math, and Congress responded a year later with the National Defense Education Act, which increased funding for scientific and technical education. The Federal Government should consider launching a similar "revolution" to emphasize skills development and prevention as the best solution to many cyber problems. This revolution could start with increased investments in programs operating under the Elementary and Secondary Education Act (K–12 programs), the Vocational Education Act (with a focus on community college curricula), and the Higher Education Act.

4.3.3 Additional U.S. Government Engagement to Advance CS Education

The U.S. government has several avenues of engagement with higher education institutions, notably the U.S. Department of Education, the National Science Foundation, and NIST. These and other entities could leverage partnerships with school districts, colleges, and universities to improve the cyber-expertise and security awareness of their CS students in additional ways such as:

- **Red teaming/blue teaming**. A computer science professor at Stanford University required his students to attack one another's code to help instill security knowledge and awareness among his students. (Note that this was not a computer security class.) Code will be attacked, successfully in many cases, so "defenders" need to understand how "attackers" think.

- **Reinforcing secure development requirements**. Include measuring security as part of subsequent CS classes to build on and reinforce the importance of secure development (as in, part of a grade is the security-worthiness of any code a student writes for the class). Without that reinforcement, security (if taught at all) is a discrete class rather than an underlying ethos. For example, the University of Virginia engineering program integrated communication into its engineering program by requiring that engineering lab classes include grades by the English department on the clarity of student writing. In addition, the English department determined half the grade on the required undergraduate thesis. This approach made communication effectiveness integral to strong engineering practice, with clarity and efficacy of communication reinforced by the grading system. A similar approach to secure development could help students understand that security must be built in, not bolted on.

- **A cyber-competency examination upon graduation**. Engineering graduates who wish to pursue a professional engineers' certification typically take an examination as part of that process. A universal competency exam in computer science (in particular, security) would reinforce the importance of security as well as qualify graduates for key positions in industry. Most suppliers would prefer to hire graduates with security knowledge rather than having to train them in the fundamentals of security assurance.

4.4. Regulatory Systems and Requirements

Malicious cybersecurity incidents are on the rise. The malware that infects systems, and any confidential information that is accessed, stolen, or used by an adverse party, cross not just computer networks but national boundaries.

The global attention to the attacks over the past year has prompted a rush by national governments to legislate or regulate. In some cases, these actions are initial responses by governments; and in others, notably in developed countries such as the U.S., new forms of legislation or regulation at the national, state, and local levels are layered on top of an existing legal infrastructure. In addition, cybersecurity laws and regulations in many jurisdictions intersect with other legal obligations in areas such as privacy, corporate governance, and labor laws. Also, current—or proposed—cybersecurity laws and regulations may be sector-specific, particularly in critical infrastructures, and may complement or conflict with sector-specific voluntary standards. In short, while the software supply chain itself is global in nature, governmental responses to malicious incidents that impact that supply chain are not global in reach or purpose.

Governments can further complicate the situation by pursuing vertical, sector-specific solutions within their national regulatory frameworks. Governments consider that specific sectors (e.g., public safety, financial services, telecommunications, heath care, energy, and IT) have unique issues that necessitate unique expertise and regulatory approaches. The result is a siloed, vertical model. Within the U.S. Government, for example, the Treasury Department regulates financial institutions, the Energy Department regulates power companies, and the Securities and Exchange Commission is exploring new regulatory requirements on publicly traded companies.

The overlapping, inconsistent regulatory frameworks can become so cumbersome that organizations divert resources toward compliance (and proof of compliance), and away from measures and approaches that might more significantly increase security posture.

In many cases, Government regulatory regimes have some elements in common, notably in:

- Establishing some measure of accountability.
- Implementing risk-based approaches.
- Identifying certain systems, such as in critical infrastructure, that should be governed under specific security measures.
- Monitoring information systems, with appropriate escalation and response measures in the event of an incident.

These common elements are foundational, but national regulatory regimes often take different approaches to achieve them. As more legislative and regulatory bodies look at ways to govern software supply chains, policymakers must weigh which measures can best advance supply chain security.

Instead of a regulatory requirement, EO 14028[81] is the Biden Administration's effort to leverage the Federal Government's buying power of information and communications technology to improve the overall security and integrity of the software supply chain. Setting standards for software assurance in the ICT products the Government buys will result in an improved software supply chain for critical infrastructure. In addition, Congress is showing a degree of bipartisan interest that has not been seen in almost a decade to impose cybersecurity standards on infrastructure ICT supply chains more directly.

Outside of the United States, the European Union is updating the Network Information Security Directive[82] to drive member-state cybersecurity strategies focused on critical infrastructure resilience against malicious attacks, including ransomware. Their approach is similar to the foundational focus on risk management, though specifics may differ across jurisdictions. The same is true in Japan, which unveiled a new cyber strategy in July 2021 that emphasized value chain trustworthiness in the security products and services provided to government and critical infrastructure.

These examples speak to the need for the U.S. Government to establish common ground by reinforcing the foundational elements in many regulatory regimes. This effort could start in two ways:

- It could establish a high-level security baseline for software assurance practices managed by a single regulator, such as DHS, with support from NIST and industry in establishing this baseline.
- It could reduce the complexity of vertical regulations by pursuing sector-specific approaches as complementary and supplemental to the security baseline.

Reinforcing foundational elements in these ways positions the U.S. Government to pursue international harmonization through multilateral approaches.

4.5. Standards Influence, Development, and Evolution

Standards are a cornerstone of software assurance approaches and processes. Continued development and alignment are required in this area. This subsection focuses on standards organizations and their complex role and influence in the field of software assurance.

Across industry, levels of maturity in software assurance and SCRM vary, with increasing interdependency across these different maturity levels. Organizations at all levels of maturity have introduced cascading risks through software supply chains. Fortunately, standards remain an integral component in enhancing security to reduce

---

[81] EO 14028, https://www.whitehouse.gov/briefing-room/presidential-actions/2021/05/12/executive-order-on-improving-the-nations-cybersecurity/

[82] ENISA, "Network Information Security Directive," https://www.enisa.europa.eu/topics/nis-directive

variability and mitigate these risks, playing a vital role in maturing software assurance across industry and the Federal Government.

However, encouraging standards in software assurance is hindered by overlapping, inconsistent, or outdated existing standards, frameworks, and guidance documents. These existing materials are often dense and their actionable elements difficult to discern, especially for users with few resources and low security maturity, such as entrepreneurial developers or small and medium-sized businesses. Current supply chain reference documents, while comprehensive, also require regular, resource-intensive updates to reflect ever-shifting dependencies, threats, and practices.

NIST introduced supply chain security controls into SP 800-53, Revision 5, *Security and Privacy Controls for Information Systems and Organizations*.[83] Along with NIST SP 800-161, *Supply Chain Risk Management Practices for Federal Information Systems and Organizations*,[84] these documents reflected a multiyear effort to develop the next generation of security and privacy controls to strengthen and support the Federal Government and critical infrastructure supply chain security. Both efforts advanced security assurance efforts especially for U.S. Government and critical infrastructure, but detailed guidance on security assurance practice for software development was beyond the scope of these efforts.

In 2020, recognizing a widening gap in security standards and software assurance practices, NIST published the SSDF for both Government and industry.[85] The SSDF helps organizations analyze and document secure software development practices while defining future practices as part of its continuous improvement process. A key component of this software assurance standard is the curated focus for both software producers (e.g., commercial-off-the-shelf vendors, Government software developers, custom software developers) and software consumers (Federal Government agencies and other organizations). NIST's SSDF is an example of a leading software assurance standard that can be integrated into existing software development workflows, products, and automated toolchains regardless of organization sector or software assurance maturity level.

The 2021 National Defense Authorization Act (NDAA), Section 882, *Balancing Security and Innovation in Software Development and Acquisition,* links software assurance and Federal Government procurement requirements. This section requires DoD software providers to follow a secure development lifecycle practice. EO 14028 further advances industry software assurance across new critical software domains, expands the use of SBOMs, and promotes adoption of zero-trust architecture. Likewise, NIST's recent definition of critical software[86] and NIST's guidance to conduct security analyses (e.g., through NIST's "Criticality Analysis Process Model:

---

[83] NIST, Joint Task Force, "NIST SP 800-53 Revision 5 "Security and Privacy Controls for Information Systems and Organizations," September 2020, https://csrc.nist.gov/publications/detail/sp/800-53/rev-5/final

[84] Boyens, Jon, et al., https://csrc.nist.gov/publications/detail/sp/800-161/final

[85] NIST SSDF, https://csrc.nist.gov/Projects/ssdf

[86] NIST, "Critical Software Definition," https://www.nist.gov/itl/executive-order-improving-nations-cybersecurity/critical-software-definition

Prioritizing Systems and Components"[87]), are valuable steps in prioritizing secure software development across the Federal Government.

Increased guidance is valuable to enhance the overall software assurance ecosystem, but future standards or revisions should remain flexible and adaptable to allow software assurance to evolve and align with modern development practices. In tandem, industry should continue implementing leading software assurance practices (e.g., SSDF) across its software lifecycles.

5. Findings and Recommendations

Software plays an increasingly significant role in national security and emergency preparedness. For example, technologies such as software defined networking have accelerated the innovation and resilience of the global communications infrastructure.

Given the increasing use of and dependence on software in critical infrastructure, this report identifies several areas for urgent action. To address these areas, **the President should establish a task force charged with defining a public-private initiative focusing on key areas of software assurance and the software supply chain. Like the earlier public-private effort on the NIST CSF,[88] such an initiative can address fundamental misalignment of incentives, diversity of the assurance approaches, and complexity of the software supply chain. An effort of this nature can translate the urgent need for action into an implementable framework.** The task force should include workstreams to define and help execute the recommendations below requiring public-private partnership as well as software assurance and SCRM expertise.

5.1. Software Assurance: Findings and Recommendations

5.1.1. Findings

No single software security assurance approach works for all situations and environments.

- No single set of software assurance practices can address every situation, due to the diversity of computing environments and development and deployment practices.
- Standards and frameworks can guide organizations, but each organization needs to tailor the best solution based on standard components and best practices adapted to the optimal approach for the organization in question.
- While there is no single approach for all environments, establishing a high-level baseline for software assurance practices while reducing the complexity of sector-specific regulations is likely to improve the

[87] NIST, "NIST IR 8179 Criticality Analysis Process Model: Prioritizing Systems and Components," https://csrc.nist.gov/publications/detail/nistir/8179/final

[88] NIST, "Framework Documents," https://www.nist.gov/cyberframework/framework

efficiency of adaptations the organizations need to make when defining their software assurance approaches.

Best practices in SCRM are not generally tailored to software.

- While the software industry has ready access to standards (e.g., ISO/IEC 20243) and best practices (e.g., NIST SP 800-161) covering SCRM and software assurance practices, these standards and practices need better adaptation to software and modern software development and deployment models.
- Adoption of these practices in both public and private sectors has been uneven.

OSS is not inherently less secure than closed source software, but incentives to invest in securing open source are neither effective nor sufficient.

- Open source software, which provides components for virtually all software products, thrives on diversity of contributions and contributor motivations. Not all contributors are motivated to adopt security assurance practices.
- Developers and administrators may not have insight into the level of security assurance for OSS modules.
- Various promising efforts are underway that may lead to improved trustworthiness and increased confidence for integrators and users of software products that contain open source. The prospects for success and impact of these efforts are still uncertain.

5.1.2. Recommendations

1.1 The Government and industry must collaborate on broader, actionable adoption of well-established, existing SCRM practices adapted to the modern software ecosystem.

a. Fund NIST to work with industry to identify, evaluate, and measure the effectiveness of new security assurance practices, including:

i. Mapping current procurement efforts and, with industry collaboration, identifying ways to promote security (e.g., based on the NIST SSDF).

ii. Completing revisions and updates already underway to SP 800-161,[89] including refining options, approaches, and requirements for SBOMs as called for in EO 14028.[90]

[89] Boyens, Jon, et al., https://csrc.nist.gov/publications/detail/sp/800-161/final

[90] EO 14028, https://www.whitehouse.gov/briefing-room/presidential-actions/2021/05/12/executive-order-on-improving-the-nations-cybersecurity/

iii. Identifying and mitigating inherited vulnerabilities from legacy infrastructure.

iv. Developing, in collaboration with industry and academic stakeholders, more robust guidance on applying cost-benefit analyses to better inform investments in software assurance across the SSDL.

v. Incentivizing industry to use external assessments where appropriate.

b. Develop and adapt standards and best practices for secure build environments for software. Tier these practices to allow appropriate scaling based on the criticality of the software and the size of the development organization. One such set of practices is the open source SLSA.

c. Examine processes used by organizations focusing on cybersecurity and software assurance, approaches in other industry sectors (e.g., telecommunications) to improve organic best practices in software assurance and the ICT supply chain.

d. Groups and task forces formulating software assurance requirements should stipulate that the diversity of developer organizations including open source organizations needs to be adequately represented. Reference DHS ICT SCRM Task Force efforts as a baseline to assess threat mitigation relative to software assurance, and task this group to provide sector-specific implementation guidance of EO 14028 directives.

1.2 Direct NIST to convene a public-private effort to improve harmonization among standards, guidelines, and frameworks in security assurance.

a. Identify gaps, conflicts, overlaps and obsolescence in software security assurance standards and frameworks.

b. Use the interagency process, public-private partnership, and global leadership to support and leverage relevant efforts, such as the NIST CSF and SSDF

c. Update the NIST CSF to refer to SSDF practices that address the capabilities gaps identified during the efforts mentioned above.

1.3 The Government should invest in R&D for the software assurance field to keep up with advances in computing architectures

a. Support R&D in software assurance in Government agencies and labs, academic research programs, and in industry to address future computing architectures.

b. Invest in innovation to automate software assurance tasks, including auditing, testing, collecting requirements, developing threat models, generating secure code, and software SCRM, specifically:

i. Increasing efficiencies for security assurance, automation, and analysis for threat modeling.

ii. Collecting and publishing representative models and patterns of coding best practices.

iii. Considering how to apply security innovations from the communications sector to other software assurance contexts; for example, developing a telecommunications top-end, common weakness enumeration list for the software development process.

c. Strengthen emerging approaches in software assurance, such as using AI and evidence-based data-driven metrics.

1.4 Improve security and assurance processes for OSS.

a. Incentivize collaboration between open source developers and organizations focusing on open source security, such as OpenSSF.

b. Task NIST to extend efforts from its work on defining critical software related to EO 14028 to identify the top open source packages used for "critical software."

c. Task the Federal Government to engage with other organizations, allied nations, and Government agencies outside of the U.S. (e.g., the European Union Agency for Cybersecurity (ENISA), the G7, or the United Nations), to create and fund a public-private software assurance program to improve open source security.

d. Develop standards to accurately describe software components, in collaboration with organizations such as OpenSSF and international standards bodies.

e. Encourage developers to adopt a system of code vetting, such as OpenSSF's Scorecard 2.0.[91]

5.2. Stakeholders: Findings and Recommendations

5.2.1. Findings

Stakeholders in development, procurement, and administration of software have different requirements and needs that are sometimes in tension.

- The software supply chain frameworks and best practices need to align with the needs of developers (who must innovate rapidly to compete), procurement teams (who must evaluate alternatives against schedule, cost, and performance requirements), and administrators (who must manage and update deployed technology.)

It is difficult to provide provable evidence of software security assurance practices.

[91] Mertic, 2021, https://openssf.org/press-release/2021/07/28/open-source-ecosystem-gains-new-support-for-securing-the-worlds-most-critical-and-pervasive-software/

- Efforts are underway to improve transparency of software assurance, but it remains challenging for stakeholders to provide useful evidence of adherence to practice.

Guidelines for software supply chain assurance are not evolving fast enough.

- The nature of software development has fundamentally changed over time and will continue to change. Cloud-based software, more frequent releases and updates, increased use of proven code modules, and the breadth of open source have all impacted how stakeholders develop, purchase, deliver, and manage software.

- Assurance frameworks must also evolve. Keeping pace with evolving technology is essential for building viable software assurance requirements.

5.2.2. Recommendations

2.1 Incentivize engagement among all stakeholders in software assurance programs, at both the domestic and international levels.

a. Engage all software lifecycle stakeholders (developers, administrators, integrators, procurement, and others) in developing adequate assurance programs with clear guidance and associated best practices.

b. Make standardization of software assurance an international effort, as companies are challenged to support an array of different requirements from multiple geographies.

c. Partner with international standards development organizations and maintain transparent operations in these organizations.

2.2 Incentivize flexible, easy-to-adopt software assurance practices for developers and suppliers.

a. Require procurement teams to prefer products that address an agency's threat models and adhere to security standards

b. Encourage the U.S. Government and industry experts to engage in global standards organizations to help keep standards global, flexible, and easy to adopt, to support the breadth of software suppliers (from small and medium businesses to large corporations).

c. Promote developer use and adoption of comprehensive software assurance practices for operations, hardware, storage, design, coding, and communications security.

d. Help document requirements for comprehensive software assurance programs, spanning threat analysis (i.e., from coding to integration, from the build environment to update mechanisms), vulnerability identification and tracking, ongoing penetration testing, build verification, and attestation (e.g., integrity checking, SBOM).

e. Continue to promote adoption of approaches such as the NIST CSF and SSDF.

2.3 Reform and update U.S. Government acquisition regulations to drive better SCRM practices, especially for designated "critical software."

a. Require procurement teams to prefer critical software (as defined by NIST in response to EO 14028[92]) that has been developed and will be maintained according to supply chain risk management frameworks such as NIST SP 800-161 or SSDF. For example, measures like the following could create new incentives to increase adoption of software security assurance practices:

   i. Rewarding vendors who invest in software assurance. For example, use gate questions in the procurement process to reinforce security as a profit center. Requiring security and SCRM as a foundational feature and functional requirements of Government procurement and tenders will incentivize vendors to prioritize security capabilities and SCRM for their offerings.

   ii. Preferring vendors who can describe their security assurance efforts in terms of existing frameworks such as NIST's SSDF.

b. Establish and support pilot programs around software component visibility and supplier evaluation tools. Activities could include:

   i. Encouraging Government procurement teams to pilot supplier evaluation tools, such as vendor templates from DHS's SCRM Task Force (e.g., the SCRM vendor template[93]).

2.4 Improve software administrator information sharing practices to increase awareness of and mitigate risks to software in use.

a. Identify and resolve current legal, procedural, and personnel challenges to timely information sharing of data on adversary activities and vulnerabilities. For example, legislation may be required to address potential liability for corporations associated with information sharing and improve incentives for information sharing in key areas.

b. Authorize specific agencies to expand use of warning systems, educational programs, and document best practices and key lessons learned.

c. Provide a single point of contact within the U.S. Government (e.g., CISA within DHS) for industry information sharing to avoid overhead and inefficiencies to industry of multiple information sharing channels.

[92] NIST, "Critical Software Definition," https://www.nist.gov/itl/executive-order-improving-nations-cybersecurity/critical-software-definition

[93] ICT SCRM Task Force, "Vendor SCRM Template," https://www.cisa.gov/publication/ict-scrm-task-force-vendor-template

d. Support incident response readiness at the federal and state levels by fostering deeper collaboration between federal, state, and national guard cyber resources and local Information and Communications Technology and Services (ICTS) cybersecurity providers.

5.3. External Influencing Factors: Findings and Recommendations

5.3.1. Findings

The global range of suppliers makes it challenging for governments to implement "one-size-fits-all" regulatory requirements.

- The economics of the software industry are unique, and components of critical software may come from suppliers in large corporations, small and medium-size businesses, or even unknown sources contributing to open source repositories.

- For critical infrastructure, sector-specific implementation guidance should be considered.

Security assurance practices are not being taught early, consistently, or broadly enough.

- Computer science and software engineering programs at universities lack consistent requirements on security assurance, and new developers frequently lack skills in security assurance practices.

- The lack of both depth and consistency requires employers to invest in costly training programs in security assurance and results in a lack of harmonization in skills and expertise, which is especially pronounced in small-medium businesses and in new areas of technology. Especially for smaller employers and startups, this burden diverts resources from developing new features, which increases competitive risk.

- While formal training in security assurance is expected to be obtained beyond the K–12 level, introducing cyber security education earlier, during K–12, could increase the numbers of future security experts and the security acumen of citizens in general.

5.3.2. Recommendations

3.1 Task the Government to create a task force to define viable incentives to support assurance practices in the extremely diverse software ecosystem.

a. Appoint a public-private representative task force to evaluate and propose economic incentives to improve software assurance and software supply chain security.

b. Invest in research on the economic aspects of software development, deployment, and administration.

c. Avoid measures that might hinder technology innovation.

3.2 Harmonize requirements for software security assurance among engineering students and in training programs.

a. Appoint a panel of higher education, industry and open source community leaders, and training organizations (e.g., ISACA[94], ISSA[95], (ISC)2[96], and SANS[97] and similar efforts outside of the U.S. such as the Cyber Security Body of Knowledge (CyBoK)[98] in the UK) with the charter to report, within one year, on recommended core security curricula for software engineering and computer science departments.

b. Incentivize inclusion of the core security curricula as graduation requirements for students enrolled in undergraduate programs in computer science.

c. Work with professional groups and security training organizations to establish postgraduate competency examinations (much like the professional engineer licensing exam for engineering graduates) and to align certifications against new requirements.

3.3 Encourage introduction of security concepts in K–12 education.

a. Encourage states to introduce security education in K–12 curricula.

6. Conclusion

Recent software supply chain compromises highlight critical risks and large-scale ramifications for industry and government. With software at the foundation of nearly every interaction in today's society, compromises in the software supply chain, especially for "critical software," can affect multiple operations of daily life.

The subcommittee examined the extremely complex area of software assurance and the software supply chain, assisted by expert briefers from industry, academia, government, and nonprofit organizations focusing on security and assurance. The diversity of approaches to software assurance and supply chain spans multiple fields and involves many organizations and individuals that use different processes to develop and update software. Even broadly adopted approaches, such as SSDL, standards (e.g., ISO/IEC 27034) and best practices (e.g., NIST's SP 800-161), are implemented differently in different organizations and supply chains. Moreover, the software lifecycle processes are inherently global.

The subcommittee concluded that:

[94] Information Systems Audit and Control Association, https://www.isaca.org

[95] Information Systems Security Association, "Cyber Security Career Lifecycle," https://www.issa.org/cyber-security-career-lifecycle/

[96] Information Security Certifications, "Earn Your Cybersecurity Certification," https://www.isc2.org/Certifications

[97] SANS, https://www.sans.org

[98] CyBOK, https://www.cybok.org/

- No single approach to software assurance and supply chain will work in all situations for all organizations and environments.
- A well-developed body of best practices and standards already exists, but its adoption is uneven.
- Several unresolved problems exist in security education and training for software development experts and IT procurement professionals, at the university level and in other settings.
- While a certain level of additional automation and new approaches to secure development should be investigated to successfully evaluate massive code bases, it is necessary to further strengthen this R&D focus area to achieve results faster.
- For an applied topic of this nature, it is necessary to engage all stakeholders – government, academia, and industry.

The President has an important role to play in improving the trustworthiness of software supply chains. President Biden's May 2021 EO 14028 represents an important step forward, laying out multiple initiatives and essential steps. NSTAC hopes this document can augment EO 14028 by identifying further opportunities for advancement.

The software assurance and software supply chain are well-developed but diffuse fields, and they will benefit from a harmonizing effort that addresses the urgency of the situation in clear, practical terms. This effort can start with establishing a task force charged with defining a broad initiative for software assurance similar to the effort that led to the creation of the NIST CSF.

Appendix A. Threat Table

Threat	Description
Compromise of Build System	Exploitation of one or more systems used as part of the overall build process that allows privileged access by an unauthorized entity to software assets used or assessed by the system.
Compromise of Code Repository	Attack against a repository meant to make modifications to source code and then push those modified files to clients.
Compromise of Deployment System	Exploitation of computing systems used to distribute final products to end consumers, allowing potential delivery of malicious or vulnerable versions of the software product.
Compromise of Design Documentation	Intentional alteration of a design document meant to introduce a weakness or vulnerability into the final product.
Compromise of Development Systems or Network	Exploitation of a development system or network component designed to provide privileged access software assets.
Compromise of Development Tools	Exploitation of design tools used in the development process that results in undetected, malicious alterations to the anticipated output.
Compromise of Requirements Documentation	Intentional alteration of a requirements documents meant to reduce the overall security of the final product, either through lack of detection or sufficient protection.
Compromise of Signing Keys	Unauthorized access to cryptographic material used as part of the signing of software products or updates.
Compromise of Test Equipment or Tools	Exploitation of equipment or tools used as part of the test process that could result in false test results or unauthorized alterations of the test candidate.
Compromise of Update System	Exploitation of computing systems used to support updates for existing products to end consumers, allowing potential delivery of malicious or vulnerable updates that could compromise the software product.
Deletion of Data	Unauthorized deletion of code, assets, or other data that results in intentional harm to the owning company or its customers.
Disable or Bypass Testing	Unauthorized removal of test requirements for software products.
Exfiltration of Source Code or Data	Extraction of original source code or other asset to an unauthorized party, either inside or outside the organization.

Extraction of Customer Information	Unauthorized access to data that may be used to identify customers of software products, including information such as company name, geographical location, quantity, or software version being used.
Falsification or Compromise of User Credentials	Unauthorized creation or use of user credentials that provide an attacker with privileged access to software assets.
Impersonate Library Repository	Creation of an illegitimate repository meant to mimic an expected repository with the intention of distributing malicious or vulnerable versions of original libraries or source code.
Injection of Malicious or Vulnerable Library	Unauthorized addition or modification of a software library that contains an intentional weakness or malicious code meant to allow exploitation of the final product.
Insider Threat	The potential malicious actions of a privileged individual at any stage of the software development lifecycle. "Privileged" refers to physical or logical access to one or more assets of the software above and beyond what a consumer of the end product may possess.
Malicious Insertion of Unauthorized Code	Unauthorized addition of malicious or vulnerable code into an existing product that lacks sufficient authentication protection.
Malicious Modification of Source Code	Intentional changes to original source code files meant to introduce a weakness or vulnerability into the final product.
Malicious Plugin for Development Tools	Distribution of a nefarious or vulnerable plugin used by a design tool that, upon installation, could make the host system susceptible to attack or exploitation.
Malicious Use of Signing Keys	Unauthorized usage of cryptographic material to sign unofficial versions of software products or updates.
Modification or Falsification of Test Results	Unauthorized removal or alteration of reports detailing the outcome of various tests meant to confirm the functionality or security of a software product.
Modification of Submission Logs	Unauthorized alteration of submission logs on repositories or shared resources meant to hide nefarious activities.
Modification of Third-Party Product	Unauthorized modification of a valid third-party product from its original form meant to introduce a weakness or vulnerability into the final product.
Modification or Poisoning of Build Process	Malicious alterations or additions to existing build processes, scripts, or tools that allow for unauthorized changes to final products.
Replacement of Valid Binaries or Patches	Unauthorized replacement of approved software binaries or patches on a respective deployment system with a malicious or vulnerable version.

Trojan Third-Party Product	Undetected malicious software module from a third party meant to be integrated into the final product of a company.

Appendix B. Government Assurance Programs: Lessons Learned

Lessons Learned from Assurance Programs

A challenge common to assurance measures is that they take considerable effort to design, deploy, and enforce. Especially in rapidly evolving technology fields, the cycle time for designing, deploying, and enforcing new assurance measures can fail to keep pace with developments in technology. Assurance programs tend to operate on the premise to industry by the U.S. Government that "if you build it in this way, we will buy it." Consequently, businesses risk spending significant resources building systems that Government customers ultimately reject.

The U.S. Government has engaged in several previous efforts to mandate improved software assurance. Given that industry and Government devote significant resources to these programs, it is important to consider lessons learned from prior efforts to inform future activities.

The Orange Book

The Orange Book was an effort led by the National Security Agency (NSA) National Computer Security Center (NCSC) in the 1980s and 1990s to define specific assurance levels and prompt industry to build systems that would meet them.[99] The Orange Book included not only specific security-related functional requirements (e.g., discretionary access control and mandatory access control), but also verification requirements (formal security evaluations by NCSC) to verify a supplier's assurance claims.

The Orange Book was one of several country-specific assurance approaches[100] that the Common Criteria (ISO/IEC 15408)[101] superseded. Common Criteria itself has undergone several modifications, including a divergence between country-specific approaches that was reconciled, to a certain extent, in the latest version.

National Security Telecommunications and Information Systems Security Policy #11

The National Security Telecommunications and Information Systems Security Policy (NSTISSP) #11 requires software used in national security systems to obtain (relevant) formal security evaluations, such as a Common Criteria evaluation.[102] In practice, this procurement requirement is often waived. As a result, suppliers who invested in meeting NSTISSP #11 requirements were unable to differentiate products and recover the cost.

---

[99] Lipner, Steven, "The Birth and Death of the Orange Book," June 2, 2015, https://ieeexplore.ieee.org/document/7116444

[100] The United Kingdom had a separate set of assurance requirements, the Information Technology Security Evaluation Criteria

[101] Common Criteria, https://www.commoncriteriaportal.org/

[102] NIAP, "Information Assurance Leadership for the Nation," March 24, 2005, https://www.niap-ccevs.org/NIAP_Evolution/faqs/nstissp-11/

Appendix C. Membership and Participants

Table 11: Subcommittee Leadership

Name	Organization	Role
Mr. Patrick Gelsinger	Intel Corp.	Subcommittee Chair
Mr. Thomas Quillin	Intel Corp.	Working Group Co-Lead
Dr. Claire Vishik	Intel Corp.	Working Group Co-Lead

Table 12: Subcommittee Membership

Name	Organization
Mr. Christopher Anderson	Lumen Technologies, Inc.
Dr. Matthew Areno	Intel Corp.
Mr. Jason Boswell	Ericsson, Inc.
Mr. Jon Boyens	National Institute of Standards and Technology
Mr. Christopher Boyer	AT&T, Inc.
Mr. Jamie Brown	Tenable, Inc.
Ms. Kathryn Condello	Lumen Technologies, Inc.
Ms. Mary Ann Davis	Oracle Corp.
Ms. Cheryl Davis	Oracle Corp.
Mr. Jon Goding	Raytheon Technologies Corp.
Mr. Dilip Gokhale	Lockheed Martin Corp.
Mr. Robert Hoffman	Broadcom, Inc.
Mr. Kent Landfield	McAfee Corp.
Mr. Coleman Mehta	Palo Alto Networks, Inc.
Mr. Sean Morgan	Palo Alto Networks, Inc.
Mr. Richard Mosley	AT&T, Inc.
Mr. Anand Pashupathy	Intel Corp.
Mr. Thomas Patterson	Unisys Corp.
Mr. John Schiel	Lumen Technologies, Inc.
Ms. Jordana Siegel	Amazon Web Services, Inc.
Mr. Chelsea Smethurst	Microsoft Corp.
Mr. Robert Spiger	Microsoft Corp.
Mr. David Stewart	Intel Corp.
Mr. Charles Taylor	Raytheon Technologies, Corp.

Mr. Sreenidhi Tummala	Lockheed Martin Corp.

Table 13: Briefers, Subject-Matter Experts

Name	Organization
Mr. Jason Boswell	Ericsson, Inc.
Mr. Jon Boyens	National Institute of Standards and Technology
Dr. Eric Brewer	Google, LLC
Mr. Randall Brooks	Raytheon Technologies Corp.
Mr. Timothy Brown	SolarWinds, Inc.
Ms. Edna Conway	Microsoft Corp.
Mr. Paul Gray	SolarWinds, Inc.
Mr. Jeffrey Greene	National Security Council
Mr. Robert Hoffman	Broadcom, Inc.
Mr. Joe Jarzombek	Synopsys Inc.
Dr. Matthew Kraning	Palo Alto Networks, Inc.
Mr. Kent Landfield	McAfee Corp.
Ms. Kim Lewandowski	Google, LLC
Mr. Steven Lipner	SAFECode
Ms. Valecia Maclin	Microsoft Corp.
Dr. John Manferdelli	VMware Inc.
Mr. Robert Martin	MITRE Corp.
Mr. Ryan Orr	Cybersecurity and Infrastructure Security Agency
Mr. Thomas Patterson	Unisys Corp.
Ms. Natalie Pittore	National Security Agency
Mr. Robert Salvia	Fortress Information Security Government Solutions
Mr. Matthew Scholl	National Institute of Standards and Technology
Mr. Brian Scott	National Security Council
Dr. Fred Schneider	Cornell University
Rear Admiral (Ret.) David Simpson	Virginia Polytechnic Institute and State University
Ms. Angela Smith	National Institute of Standards and Technology
Ms. Kate Stewart	Linux Foundation
Mr. Bryan Ware	Next5, Inc.
Mr. Robert Walters	Symantec Corp.
Mr. Theodore Winograd	Booz Allen Hamilton, Inc.

Dr. David Wheeler	Linux Foundation

Table 14: Subcommittee Management

Name	Organization
Ms. Sandra Benevides	President's National Security Telecommunications Advisory Committee (NSTAC) Designated Federal Officer (DFO)
Ms. DeShelle Cleghorn	Alternate NSTAC DFO
Mr. Scott Zigler	NSTAC Program Support
Ms. Sheila Becherer	Booz Allen Hamilton, Inc.
Ms. Emily Berg	Booz Allen Hamilton, Inc.
Dr. Philip Grant	Booz Allen Hamilton, Inc.
Ms. Laura Penn	Insight Technology Solutions, LLC

Appendix D. Acronyms

Acronym	Definition
5G	Fifth Generation
AI	Artificial Intelligence
ANSI	American National Standards Institute
ANSDIT	American National Standard Dictionary of Information Technology
API	Application Programming Interface
BSA	Business Software Alliance
BSIMM	Building Security in Mature Model
CI/CD	Continuous Integration/Continuous Deployment
CISA	Cybersecurity and Infrastructure Security Agency
CNSS	Committee on National Security Systems
CNSSI	Committee on National Security Systems Instruction
COVID-19	Coronavirus Disease 2019
CS	Computer Science
CSF	Cybersecurity Framework
CSP	Cloud Service Provider
CyBOK	Cybersecurity Body of Knowledge
DARPA	Defense Advanced Research Projects Agency
DAST	Dynamic Application Security Testing
DevOps	Development Operations
DevSecOps	Development, Security, and Operations
DFO	Designated Federal Officer
DHS	Department of Homeland Security
DoD	Department of Defense
DoS	Denial of Service
EO	Executive Order
EU	European Union
FAR	Federal Acquisition Regulation
FCC	Federal Communications Commission
FIPS	Federal Information Processing Standards
IaaS	Infrastructure as a Service
ICT	Information and Communications Technology
ICTS	Information and Communications Technology and Services

IEC	International Electrotechnical Commission
INCITS	International Committee for Information Technology Standards
IoT	Internet of Things
ISACA	Information Systems Audit and Control Association
ISO	International Organization for Standardization
ISP	Internet Service Provider
ISSA	Information Systems Security Association
IT	Information Technology
MFA	Multifactor Authentication
ML	Machine Learning
NCSC	National Computer Security Center
NFV	Network Function Virtualization
NIAP	National Information Assurance Partnership
NIST	National Institute of Standards and Technology
NISTIR	National Institute of Standards and Technology Interagency or Internal Report
NPM	Node Package Manager
NRMC	National Risk Management Center
NSA	National Security Agency
NSC	National Security Council
NS/EP	National Security and Emergency Preparedness
NSTAC	National Security Telecommunications Advisory Committee
NSTISSP	National Security Telecommunications and Information Systems Security Policy
NTIA	National Telecommunications and Information Administration
ODNI	Office of the Director of National Intelligence
OpenSSF	Open Source Security Foundation
OpenSSL	Open Secure Sockets Layer
OSS	Open Source Software
OT	Operational Technology
OWASP	The Open Web Application and Security Project
PaaS	Platform-as-a-Service
RF	Radio Frequency
SaaS	Software-as-a-Service
SBOM	Software Bill of Materials
SCRM	Supply Chain Risk Management

SDL	Software Development Lifecycle
SLSA	Supply Chain Levels for Software Artifacts
SME	Subject Matter Expert
SP	Special Publication
SSDF	Secure Software Development Framework
SSDL	Secure Software Development Lifecycle
U.S.	United States
V&T	Verification and Testing
TLS	Transport Layer Security
ZTA	Zero-Trust Architecture

Appendix E. Definitions

Term	Definition	Source
Adversary	Any individual, group, organization, or government that conducts or has the intent to conduct detrimental activities.	• National Institute of Standards and Technology (NIST) Special Publication (SP) 800-30
Application Programming Interface	A system access point or library function that has a well-defined syntax and is accessible from application programs or user code to provide well-defined functionality.	• NIST SP 1800-16C under "application program interface" from NIST Interagency or Internal Report (NISTIR) 5153
Artificial Intelligence	(1) A branch of computer science devoted to developing data processing systems that performs functions normally associated with human intelligence, such as reasoning, learning, and self-improvement. (2) The capability of a device to perform functions that are normally associated with human intelligence such as reasoning, learning, and self-improvement.	• American National Standards Institute International Committee for Information Technology Standards (INCITS) 172-220 (R2007) Information Technology -- American National Standard Dictionary of Information Technology (ANSDIT) • Cited in NIST's *U.S. Leadership in AI: A Plan for Federal Engagement in Developing Technical Standards and Related Tools*
Broadband	High-speed internet access that is always on and faster than traditional dial-up access.	• Federal Communications Commission (FCC), https://www.fcc.gov/general/types-broadband-connections#:~:text=The%20%20term%20broadband%20commonly%20refers%20to%20high-speed%20Internet,transmission%20technologies%20%20such%20as:%20Digital%20Subscriber%20Line%20
Cloud Computing	A model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction.	• NISTIR 8006 under "cloud computing"
Commercial off-the-Shelf	Software and hardware that already exist and are available from commercial sources.	• NIST SP 800-161 under "commercial off-the-shelf" NIST SP 800-64 Rev. 2
Connectivity	Capacity for the interconnection of platforms, systems, and applications.	• PCMag, https://www.pcmag.com/encyclopedia/term/connectivity

Counterfeit	An unauthorized copy or substitute that has been identified, marked, and/or altered by a source other than the item's legally authorized source and has been misrepresented to be an authorized item of the legally authorized source.	• NIST SP 800-161, 18 U.S.C.
Critical Infrastructure	Sixteen sectors whose assets, systems, and networks, whether physical or virtual, are considered so vital to the United States that their incapacitation or destruction would have a debilitating effect on security, national economic security, national public health or safety, or any combination thereof.	• Cybersecurity Infrastructure Security Agency, https://www.cisa.gov/critical-infrastructure-sectors
Cybersecurity	: Prevention of damage to, protection of, and restoration of computers, electronic communications systems, electronic communications services, wire communication, and electronic communication, including information contained therein, to ensure its availability, integrity, authentication, confidentiality, and nonrepudiation.	• Committee on National Security Systems Instruction (CNSSI) 4009-2015 from NSPD-54/HSPD-23 • NIST SP 1800-25B under Cybersecurity from CNSSI 4009-2015 • NSPD-54/HSPD-23 • NIST SP 1800-26B under Cybersecurity from CNSSI 4009-2015 • NSPD-54/HSPD-23 • NIST SP 800-160 Vol. 2 from CNSSI 4009-2015 • NIST SP 800-37 Rev. 2 • NIST SP 800-53 Rev. 5 from OMB Circular A-130 (2016) • NISTIR 7621 Rev. 1 under Cybersecurity from CNSSI 4009-2015
Denial-of-Service	The prevention of authorized access to resources or the delaying of time-critical operations. (Time-critical may be milliseconds or it may be hours, depending upon the service provided).	• NIST SP 800-12 Rev. 1 under Denial of Service from CNSSI 4009
Development Operations	A set of practices for automating the processes between software development and information technology operations teams so that they can build, test, and release software faster and more reliably. The goal is to shorten the systems development life cycle and improve reliability while delivering features, fixes, and updates frequently in close alignment with business objectives.	• NIST SP 1800-16B • NIST SP 1800-16C • NIST SP 1800-16D
Emerging Technologies	Technologies that are currently developing and are expected to impact society in some significant way over the next 5 to 10 years.	• Independence University, https://www.independence.edu/blog/what-is-emerging-technology

Executive Order 14028, *Improving the Nation's Cybersecurity*	The President's Executive Order (EO) issued on May 12, 2021, charges multiple agencies – including NIST– with enhancing cybersecurity through a variety of initiatives related to the security and integrity of the software supply chain.	• Federal Register: Improving the Nation's Cybersecurity
Fifth Generation	The fifth installment of advanced wireless technology, bringing about increased bandwidth and capacity for advancements within the Internet of Things.	• Qualcomm, https://www.qualcomm.com/5g/what-is-5g
Hardware	The physical components of an information system.	• NIST SP 800-53 Rev. 4 under Hardware CNSSI 4009
Information Technology	Any equipment or interconnected system or subsystem of equipment that is used in the automatic acquisition, storage, manipulation, management, movement, control, display, switching, interchange, transmission, or reception of data or information by the executive agency. For purposes of the preceding sentence, equipment is used by an executive agency if the equipment is used by the executive agency directly or is used by a contractor under a contract with the executive agency which: (i) requires the use of such equipment; or (ii) requires the use, to a significant extent, of such equipment in the performance of a service or the furnishing of a product. The term information technology includes computers, ancillary equipment, software, firmware and similar procedures, services (including support services), and related resources.	• Federal Information Processing Standards 200 under Information Technology 40 U.S.C., Sec. 1401
Internet of Things	Internet of Things (IoT) refers to systems that involve computation, sensing, communication, and actuation (as presented in NIST SP 800-183). IoT involves the connection between humans, non-human physical objects, and cyber objects, enabling monitoring, automation, and decision making.	• NIST SP 800-183
Internet Protocol	Standard protocol for transmission of data from source to destinations in packet switched communications networks and interconnected systems of such networks.	• CNSSI 4009-2015
Internet Service Providers	A company that provides internet connections and services to individuals and organizations.	• Britannica, https://www.britannica.com/technology/Internet-service-provider
Malware	Hardware, firmware, or software that is intentionally included or inserted in a system for a harmful purpose.	• CNSSI 4009-2015 under malicious logic from Internet Engineering Task Force Request for Comments 4949 V2
National Security and Emergency Preparedness	Policies, plans, procedures, and readiness measures that enhance the ability of the U.S. Government to mobilize for, respond to, and recover from a national security emergency.	• Department of the Interior, https://www.doi.gov/sites/doi.gov/files/-900-dm-5-nsep-2021.pdf

Operating System	The software "master control application" that runs the computer. It is the first program loaded when the computer is turned on, and its main component, the kernel, resides in memory at all times. The operating system sets the standards for all application programs (such as the Web server) that run in the computer. The applications communicate with the operating system for most user interface and file management operations.	• NIST SP 800-44 Version 2 • NISTIR 7621 Rev. 1 from NIST SP 800-44 Version 2
Operational Technology	Programmable systems or devices that interact with the physical environment (or manage devices that interact with the physical environment). These systems/devices detect or cause a direct change through the monitoring and/or control of devices, processes, and events. Examples include industrial control systems, building management systems, fire control systems, and physical access control mechanisms.	• NIST SP 800-37 Rev. 2
Protocol	A set of rules governing the exchange or transmission of data between devices.	• Britannica, https://www.britannica.com/technology/protocol-computer-science
Software Applications	A software program hosted by an information system.	• CNSSI 4009-2015 from NIST SP 800-37 Rev. 1 • NIST SP 1800-16B under Application from NIST SP 800-137 • NIST SP 1800-16C under Application from NIST SP 800-137 • NIST SP 1800-16D under Application from NIST SP 800-137 • NIST SP 800-137 under Application from NISTIR 7298 • NIST SP 800-37 Rev. 2 • NIST SP 800-53 Rev. 5 from NIST SP 800-37 Rev. 2 • NISTIR 7621 Rev. 1 under Application from CNSSI 4009-2015 • NIST SP 800-37 Rev. 1 [Superseded] under Application
Software Developers	A person, or group, that designs and/or builds and/or documents and/or configures the hardware and/or software of computerized systems.	• Food and Drug Administration, Glossary of Computer System Software Development Terminology (8/95)
Software Development Lifecycle	The scope of activities associated with a system, encompassing the system's initiation, development and acquisition, implementation, operation, and maintenance, and ultimately its disposal that instigates another system initiation.	• CNSSI 4009-2015 from NIST SP 800-34 Rev. 1

Third-Party Component	An external entity, including, but not limited to, service providers, vendors, supply-side partners, demand-side partners, alliances, consortiums, and investors, with or without a contractual relationship to the first-party organization.	• NIST, https://csrc.nist.gov/glossary/term/Third_Party_Relationships
Threat	Any circumstance or event with the potential to adversely impact agency operations (including mission, functions, image, or reputation), agency assets, or individuals through an information system via unauthorized access, destruction, disclosure, modification of information, and/or DoS.	• NIST SP 800- 53, CNSSI 4009, Adapted
Threat Environment	The online space where cyber threat actors conduct malicious cyber threat activity. ()	• An Introduction to the Cyber Threat Environment, https://icclr.org/wp-content/uploads/2019/05/Intro-to-cyber-threat-environment-e.pdf?x37853
Trustworthiness	The attribute of a person or enterprise that provides confidence to others of the qualifications, capabilities, and reliability of that entity to perform specific tasks and fulfill assigned responsibilities.	• NIST SP 800-39, CNSSI-4009
Verification	Confirmation, through the provision of objective evidence, that specified requirements have been fulfilled (e.g., an entity's requirements have been correctly defined, or an entity's attributes have been correctly presented; or a procedure or function performs as intended and leads to the expected outcome).	• NIST SP 800-161 under Verification from CNSSI 4009 • ISO 9000 – Adapted • NISTIR 7622 under Verification from CNSSI 4009, ISO 9000 – Adapted
Virtual Private Network	A virtual network built on top of existing networks that can provide a secure communications mechanism for data and IP information transmitted between networks.	• NIST SP 800-113 under Virtual Private Network
Zero-Trust Architecture	An architecture that treats all users as potential threats and prevents access to data and resources until the users can be properly authenticated and their access authorized.	• NIST, https://www.nccoe.nist.gov/projects/building-blocks/zero-trust-architecture

Appendix F. Bibliography

Acquisition.Gov, "Federal Acquisition Regulation," https://www.acquisition.gov/browse/index/far

Areno, Matthew and Martin, Antonio, "Supply Chain Threats-Software White Paper," *Intel*, July 2021, https://www.intel.com/content/www/us/en/security/supply-chain-threat-whitepaper.html

Bergin, Thomas J., "Mainframe Computers," American University, 2012, http://ds-wordpress.haverford.edu/bitbybit/wp-content/uploads/2012/07/Lecture-9-Mainframe-Computing.pdf

Boswell, Jason, Ericsson, "Overview of Department of Homeland Security Information and Communications Technology [ICT] Supply Chain Risk Management [SCRM] Task Force. SCRM Task Force Working Group 4: Vendor SCRM Template," Briefing to the President's National Security Telecommunications Advisory Committee (NSTAC) Software Assurance (SA) Subcommittee, Arlington, VA, June 15, 2021

Boyens, Jon et al., National of Institute of Standards and Technology (NIST), Special Publication (SP) 800-161: *SCRM Practices for Federal Information Systems and Organizations*, April 2015, https://csrc.nist.gov/publications /detail/sp/800-161/final

Boyens, Jon and Smith, Angela, NIST, "Cyber Supply Chain Risk Management," Briefing to the NSTAC SA Subcommittee, Arlington, VA, June 29, 2021

Brewer, Eric, Google, "Open Source Software (OSS) *is* the Critical Software," Briefing to the NSTAC SA Subcommittee, Arlington, VA, August 19, 2021

Broadcom, "Affiliates Unlocked: Gangs Switch Between Different Ransomware Families," August 12, 2021, https://symantec-enterprise-blogs.security.com/blogs/threat-intelligence/ransomware-trends-lockbit-sodinokibi

Brooks, Randall, Raytheon Technologies, "Experiences with Cyber SCRM and Training SA," Briefing to the NSTAC SA Subcommittee, Arlington, VA, September 2, 2021

Brown, Timothy and Gray, Paul, SolarWinds, "Incident Update and Next Generation Build Environment," Briefing to the NSTAC SA Subcommittee, Arlington, VA, July 22, 2021

Building Security in Maturity Model Framework, https://www.bsimm.com/framework.html

Business Software Alliance (BSA) Foundation, "BSA Framework for Secure Software," https://www.bsa.org/reports/updated-bsa-framework-for-secure-software

Champion, Kaylea and Mako Hill, Benjamin, "Underproduction: An Approach for Measuring Risk in OSS," February 27, 2021, https://arxiv.org/abs/2103.00352

Cybersecurity and Infrastructure Security Agency (CISA), "Alert (TA14-098A) Open Secure Sockets Layer 'Heartbleed' Vulnerability (Common Vulnerabilities and Exposures-2014-016)," revised October 2016, https://us-cert.cisa.gov/ncas/alerts/TA14-098A

CISA, "Remediating Microsoft Exchange Vulnerabilities," Remediating Microsoft Exchange Vulnerabilities | CISA

Clancy, Charles, Ledgett, Jr., Richard H., Nissen, Christopher A., Sledjeski, Christopher L., MITRE, "Beyond SolarWinds: Principles for Securing Software Supply Chains," March, 2021, https://www.mitre.org/publications/technical-papers/beyond-solarwinds-principles-for-securing-software-supply-chains

Collins, Keith, Quartz, "How One Programmer Broke the Internet by Deleting a Tiny Piece of Code," March 27, 2016, https://qz.com/646467/how-one-programmer-broke-the-internet-by-deleting-a-tiny-piece-of-code/

Common Criteria, https://www.commoncriteriaportal.org/

Conway, Edna, Microsoft, "ICT and Services Security and Resiliency," Briefing to the NSTAC SA Subcommittee, Arlington, VA, June 24, 2021

Cyber Security Body of Knowledge, https://www.cybok.org/

CycloneDX, "Open Web Application Security Project CycloneDX is a Lightweight Software Bill of Materials [SBOM] Standard Designed For Use in Application Security Contexts and Supply Chain Component Analysis," https://cyclonedx.org

European Commission, "The European Union Cybersecurity Act," https://digital-strategy.ec.europa.eu/en/policies/cybersecurity-act

European Union Agency for Cybersecurity (ENISA), https://www.enisa.europa.eu

ENISA, "Network Information Security Directive," https://www.enisa.europa.eu/topics/nis-directive

Executive Order (EO) 14028, *Improving the Nation's Cybersecurity*, The White House, May 12, 2021, https://www.whitehouse.gov/briefing-room/presidential-actions/2021/05/12/executive-order-on-improving-the-nations-cybersecurity/.

Federal Communications Commission (FCC), "Remarks of FCC Chairman Ajit Pai to the Center for Strategic and International Studies," January 5, 2021, https://docs.fcc.gov/public/attachments/DOC-369080A1.pdf

Federal Risk and Authorization Management Program, "Securing Cloud Services for the Federal Government," https://www.fedramp.gov/

Gilmore, Shaun, Simpson, Stacy, and Sondhi, Reeny, SAFECode, "Principles for SA Assessment," 2015, https://safecode.org/publication/SAFECode_Principles_for_Software_Assurance_Assessment.pdf

Github, "Security Levels," https://github.com/slsa-framework/slsa/blob/main/docs/levels.md

Github, "Supply-Chain Levels for Software Artifacts Framework," https://github.com/slsa-framework/slsa#readme

Github, "The 2020 State of the Octo-Verse," 2020, https://octoverse.github.com/

Goldstein, Phil, FedTech, "Defense Advanced Research Projects Agency Explores How to Automate SA Assessments," July 8, 2019, https://fedtechmagazine.com/article/2019/07/darpa-explores-how-automate-software-assurance-assessments

Google, "Securing the Software Development Lifecycle with Cloud Build and Supply Chain Levels for Software Artifacts," July 29, 2021, https://cloud.google.com/blog/products/application-development/google-introduces-slsa-framework

Gordon, Lawrence A., Loeb, Martin P., and Zhou, Lei, "Integrating the Cost-Benefit Analysis into the NIST Cybersecurity Framework via the Gordon-Loeb Model," Journal of Cybersecurity, Volume 6, Issue 1, March 30, 2020, https://academic.oup.com/cybersecurity/article/6/1/tyaa005/5813544

Greene, Jeffrey and Scott, Brian, National Security Council, "EO 14028 Impacts on NSTAC Software Assurance Study," Briefing to the NSTAC SA Subcommittee, Arlington, VA, June 22, 2021

Ham, Haylee, Lifschitz-Assaf, Hila, Nagle, Frank, Wheeler, David A., The Linux Foundation and The Laboratory for Innovation Science at Harvard, "Report on the 2020 Free/Open Source Software Contributor Survey," December 8, 2020, https://www.linuxfoundation.org/wp-content/uploads/2020FOSSContributorSurveyReport_121020.pdf

Hoffman, Robert, Broadcom, "Internal and External Influences," August 2021

ICT SCRM Task Force, CISA, "Mitigating ICT Supply Chain Risks with Qualified Bidder and Manufacturer Lists," April 2021, https://www.cisa.gov/publication/ict-scrm-task-force-qualified-lists-report

ICT SCRM Task Force, CISA, "Vendor SCRM Template," April 2021, https://www.cisa.gov/publication/ict-scrm-task-force-vendor-template

Information Security Certifications, "Earn Your Cybersecurity Certification," https://www.isc2.org/Certifications

Information Systems Audit and Control Association, https://www.isaca.org

Information Systems Security Association, "Cyber Security Career Lifecycle," https://www.issa.org/cyber-security-career-lifecycle/

International Organization for Standardization and the International Electrotechnical Commission 27034-1:2011, *Information Technology – Security Techniques – Application Security – Part 1: Overview and Concepts*, https://www.iso.org/standard/44378.html

Interos, "Interos Annual Global Supply Chain Report," June 24, 2021, https://www.interos.ai/resources/global-supply-chain-report/

Interos, "Supply Chain Disruptions and the High Cost of the Status Quo," August 3, 2021, https://www.interos.ai/resources/whitepaper-supply-chain-disruptions-and-the-status-quo/

Jarzombek, Joe, Synopsys, "Embracing SA Principles in Critical Infrastructure," Briefing to the NSTAC SA Subcommittee, Arlington, VA, July 27, 2021

Kraning, Matthew, Palo Alto Networks, "Internet Operations," Briefing to the NSTAC SA Subcommittee, Arlington, VA, July 15, 2021

Landfield, Kent et al., "Enduring Security Framework: SA and Supply Chains," Briefing to the NSTAC SA Subcommittee, Arlington, VA, June 22, 2021

Lewandowski, Kim, Google, "Open Source Supply Chain Security," Briefing to the NSTAC SA Subcommittee, Arlington, VA, August 12, 2021

Lipner, Steven, SAFECode, "Industry Approaches to Software SA and Supply Chain," Briefing to the NSTAC SA Subcommittee, Arlington, VA, May 27, 2021

Lipner, Steven, "The Birth and Death of the Orange Book," June 2, 2015, https://ieeexplore.ieee.org/document/7116444

Lynch, Sarah, "U.S. Launches Online Hub to Help Ransomware Victims," July 15, 2021, https://www.reuters.com/world/us/us-launches-online-hub-help-ransomware-victims-2021-07-15/

Maclin, Valecia, Microsoft, "Security by Design," Briefing to the NSTAC SA Subcommittee, Arlington, VA, August 5, 2021

Manferdelli, John, VMware, "SA," Briefing to the NSTAC SA Subcommittee, Arlington, VA, June 8, 2021

Martin, Robert, MITRE, "SA," Briefing to the NSTAC SA Subcommittee, Arlington, VA, July 1, 2021

Mertic, John, Open Source Security Foundation, The Linux Foundation Projects, "Open Source Ecosystem Gains New Support for Securing the World's Most Critical and Pervasive Software," July 28, 2021,

https://openssf.org/press-release/2021/07/28/open-source-ecosystem-gains-new-support-for-securing-the-worlds-most-critical-and-pervasive-software/.

Microsoft, "About Microsoft Security Development Lifecycle," 2021, https://www.microsoft.com/en-us/securityengineering/sdl/about

Microsoft, "HAFNIUM Targeting Exchange Servers with 0-day Exploits," March 2, 2021, https://www.microsoft.com/security/blog/2021/03/02/hafnium-targeting-exchange-servers

Microsoft, "The Traditional Microsoft Product Development Process, The Security Development Lifecycle," May 22, 2012, https://docs.microsoft.com/en-us/previous-versions/windows/desktop/cc307406(v=msdn.10)#:~:text=The%20Traditional%20Microsoft%20Product%20Development%20Process%20In%20response,one%20or%20two%20prior%20versions%20of%20the%20code

MITRE ATT&CK, "SUNSPOT," January 12, 2021, https://attack.mitre.org/software/S0562/

National Information Assurance Partnership (NIAP), "About NIAP," https://www.niap-ccevs.org/

NIAP, "Information Assurance Leadership for the Nation," March 24, 2005, https://www.niap-ccevs.org/NIAP_Evolution/faqs/nstissp-11/

National Telecommunications and Information Administration, "The Minimum Elements for a SBOM," July 12, 2021, https://www.ntia.doc.gov/report/2021/minimum-elements-software-bill-materials-sbom

NIST, "Computer Security Resource Center," https://csrc.nist.gov/

NIST, "Critical Software Definition," https://www.nist.gov/itl/executive-order-improving-nations-cybersecurity/critical-software-definition NIST, "EO 14028: Improving the Nation's Cybersecurity," https://www.nist.gov/itl/executive-order-improving-nations-cybersecurity/critical-software-definition

NIST "Framework Documents," https://www.nist.gov/cyberframework /framework.

NIST, "Internal Report 8179 Criticality Analysis Process Model: Prioritizing Systems and Components," https://csrc.nist.gov/publications/detail/nistir/8179/final

NIST, Joint Task Force, "NIST SP 800-53 Revision 5 "Security and Privacy Controls for Information Systems and Organizations," September 2020, https://csrc.nist.gov/publications/detail/sp/800-53/rev-5/final

NIST, National Vulnerability Database, https://nvd.nist.gov/vuln-metrics/cvss

NIST, "Open Source Code," December 6, 2018, https://www.nist.gov/document/finals610601ver1pdf

NIST, "Secure Software Development Framework," https://csrc.nist.gov/Projects/ssdf

NIST, "Software Identification Tagging," https://csrc.nist.gov/projects/software-identification-swid

Nissen, Christopher et al, MITRE, "Deliver Uncompromised: A Strategy for Supply Chain Security and Resilience in Response to the Changing Character of War," https://www.mitre.org/publications/technical-papers/deliver-uncompromised-a-strategy-for-supply-chain-security

Office of the Under Secretary of Defense for Acquisition & Sustainment, "Cybersecurity Maturity Model Certification," https://www.acq.osd.mil/cmmc/

Open Source Security Foundation, The Linux Foundation Projects, "Securing the Open Source Ecosystem," https://openssf.org/

OWASP Foundation, "OWASP Software Assurance Maturity Model," https://owasp.org/www-project-samm/

OWASP, "OWASP Software Component Verification Standard," https://owasp.org/www-project-software-component-verification-standard/

Paulsen, Celia et al., NIST, "NIST Internal Report 8179: Criticality Analysis Process Model: Prioritizing Systems and Components," April 2018, https://csrc.nist.gov/publications/detail/nistir/8179/final

Pittore, Natalie, National Security Agency, and Orr, Ryan, CISA, "Enduring Security Framework Introductions and Mitigation Efforts" and "Fifth Generation Threat Landscape," Briefing to the NSTAC SA Subcommittee, Arlington, VA, June 1, 2021

Poretsky, Scott, et al, Ericsson, "Open Source Software Security in an ICT Context – Benefits, Risks, and Safeguards," January 14, 2021. https://www.ericsson.com/en/blog/2021/1/open-source-security-software

President's NSTAC, "NSTAC Report to the President on Communications Resiliency," April 2011, https://www.cisa.gov/publication/2011-nstac-publications

Regenscheid, Andrew, NIST, "NIST Special Publication 800-193 Platform Firmware Resiliency Guidelines," May 2018, https://csrc.nist.gov/publications/detail/sp/800-193/final

SAFECode, "Fundamental Practices for Secure Software Development," March 2018, https://safecode.org/fundamental-practices-secure-software-development/

Sanger, David et al, "Cyberattack Forces a Shutdown of a Top U.S. Pipeline," May 8, 2021, https://www.nytimes.com/2021/05/08/us/politics/cyberattack-colonial-pipeline.html

Salvia, Robert, Fortress Information Security Government Solutions, "Operationalizing Cyber Supply Chain Risk Management," Briefing to the NSTAC SA Subcommittee, Arlington, VA, July 22, 2021

SANS, https://www.sans.org

Schneider, Fred, Cornell University, "Notes on Supply Chains and Trustworthiness," Briefing to the NSTAC SA Subcommittee, Arlington, VA, June 17, 2021

Scholl, Matthew, NIST, "Encryption, Trust Foundations, and Software," Briefing to the NSTAC SA Subcommittee, Arlington, VA, July 6, 2021

Security Magazine, "Synopsys Study Shows 91% of Commercial Applications Contain Outdated or Abandoned Open Source Components," May 12, 2020, https://www.securitymagazine.com/articles/92368-synopsys-study-shows-91-of-commercial-applications-contain-outdated-or-abandoned-open source-components

Simpson, David, Virginia Polytechnic Institute and State University, "Protect United States ICT Supply Chain and Preserve Global Tech Leadership," Briefing to the NSTAC SA Subcommittee, Arlington, VA, July 20, 2021

Software Package Data Exchange, "The Software Package Data Exchange," https://spdx.dev

SolarWinds Worldwide, LLC, "SolarWinds Security Advisory." April 6, 2021, https://www.solarwinds.com/sa-overview/securityadvisory#anchor1

Stack Overflow, http://stackoverflow.com

Stewart, Kate and Wheeler, David, Linux Foundation, "Open Source and Securing the Software Supply Chain," Briefing to the NSTAC SA Subcommittee, Arlington, VA, June 10, 2021

Synopsys, "Synopsys Study Shows Uptick in Vulnerable, Outdated, and Abandoned Open Source Components in Commercial Software," April 13, 2021, https://news.synopsys.com/2021-04-13-Synopsys-Study-Shows-Uptick-in-Vulnerable-Outdated-and-Abandoned-Open-Source-Components-in-Commercial-Software

The Forum of Incident Response and Security Teams, Common Vulnerability Scoring System Special Interest Group, https://www.first.org/cvss/

Tung, Liam, "Kaseya Ransomware Attack: 1,500 Companies Affected, Company Confirms," July 6, 2021, https://www.zdnet.com/article/kaseya-ransomware-attack-1500-companies-affected-company-confirms/

Verizon, "2021 Data Breach Investigations Report," https://www.verizon.com/business/resources/reports/dbir/

Ware, Bryan, Next5, "The Future of Information Technology," Briefing to the NSTAC SA Subcommittee, Arlington, VA, August 5, 2021

Weeks, Derek, "Introducing Our 2020 State of the Software Supply Chain Report," August 12, 2021, https://blog.sonatype.com/2020-state-of-the-software-supply-chain-report

Winograd, Ted, Booz Allen Hamilton, "Integrating SA into Development and Operations and Zero-Trust Architectures," Briefing to the NSTAC SA Subcommittee, Arlington, VA, August 12, 2021