SOAR

STATE-OF-THE-ART REPORT (SOAR)
JANUARY 2024

CSIAC-BCO-2023-499

APPLICATIONS OF ARTIFICIAL INTELLIGENCE (AI) FOR PROTECTING SOFTWARE SUPPLY CHAINS (SSCS) IN THE DEFENSE INDUSTRIAL BASE (DIB)

By Abdul Rahman
Contract Number: FA8075-21-D-0001
Published By: CSIAC

This Page Intentionally Left Blank

SOAR

STATE-OF-THE-ART REPORT (SOAR)
JANUARY 2024

APPLICATIONS OF ARTIFICIAL INTELLIGENCE (AI) FOR PROTECTING SOFTWARE SUPPLY CHAINS (SSCS) IN THE DEFENSE INDUSTRIAL BASE (DIB)

ABDUL RAHMAN

ABOUT CSIAC

The Cybersecurity & Information Systems Information Analysis Center (CSIAC) is a U.S. Department of Defense (DoD) IAC sponsored by the Defense Technical Information Center (DTIC). CSIAC is operated by SURVICE Engineering Company under contract FA8075-21-D-0001 and is one of the three next-generation IACs transforming the DoD IAC program: CSIAC, Defense Systems Information Analysis Center (DSIAC), and Homeland Defense & Security Information Analysis Center (HDIAC).

CSIAC serves as the U.S. national clearinghouse for worldwide scientific and technical information in four technical focus areas: cybersecurity; knowledge management and information sharing; modeling and simulation; and software data and analysis. As such, CSIAC collects, analyzes, synthesizes, and disseminates related technical information and data for each of these focus areas. These efforts facilitate a collaboration between scientists and engineers in the cybersecurity and information systems community while promoting improved productivity by fully leveraging this same community's respective knowledge base. CSIAC also uses information obtained to generate scientific and technical products, including databases, technology assessments, training materials, and various technical reports.

State-of-the-art reports (SOARs)—one of CSIAC's information products—provide in-depth analysis of current technologies, evaluate and synthesize the latest technical information available, and provide a comprehensive assessment of technologies related to CSIAC's technical focus areas. Specific topic areas are established from collaboration with the greater cybersecurity and information systems community and vetted with DTIC to ensure the value-added contributions to Warfighter needs.

CSIAC's mailing address:

CSIAC
4695 Millennium Drive
Belcamp, MD 21017-1505
Telephone: (443) 360-4600

REPORT DOCUMENTATION PAGE	*Form Approved OMB No. 0704-0188*

The public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing the burden, to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.
PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.

1. REPORT DATE January 2024	**2. REPORT TYPE** State-of-the-Art Report	**3. DATES COVERED**
4. TITLE AND SUBTITLE Applications of Artificial Intelligence (AI) for Protecting Software Supply Chains (SSCs) in the Defense Industrial Base (DIB)		**5a. CONTRACT NUMBER** FA8075-21-D-0001
		5b. GRANT NUMBER
		5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) Abdul Rahman		**5d. PROJECT NUMBER**
		5e. TASK NUMBER
		5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Cybersecurity & Information Systems Information Analysis Center (CSIAC) SURVICE Engineering Company 4695 Millennium Drive Belcamp, MD 21017-1505		**8. PERFORMING ORGANIZATION REPORT NUMBER** CSIAC-BCO-2023-499
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Defense Technical Information Center (DTIC) 8725 John J. Kingman Road Fort Belvoir, VA 22060		**10. SPONSOR/MONITOR'S ACRONYM(S)** DTIC
		11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
DISTRIBUTION STATEMENT A. Approved for public release: distribution unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT
The application of artificial intelligence (AI) to software supply chains (SSCs) within the defense industrial base (DIB) holds promise to improve cybersecurity posture, ensure stricter compliance with National Institute of Standards and Technology (NIST) controls, and increase user confidence in software built in part upon modules and libraries from outside repositories. AI can provide analysts with suggested frequencies for (re)scanning, supplement threat assessments of infrastructure, automate threat intelligence processing, and expedite cybersecurity risk management. Moreover, the security of SSCs in the DIB can benefit from similar uses of AI as a recommendation engine for communicating the probability of compromise. For U.S. Department of Defense cybersecurity analysts, AI-driven automation can provide insight into how closely software capabilities deployed on military and government networks adhere to NIST compliance standards. The ability to reflect the most up-to-date set of vulnerabilities within a system security plan could significantly improve upon the existing practice of relying on manual internal scanning. AI can enable human-in-the-loop workflows to optimize the integration of processed threat intelligence and better identify vulnerabilities per software and/or operating system. This report presents and discusses how AI can protect SSCs purpose-built for the DIB ecosystem.

15. SUBJECT TERMS
cybersecurity, cyberattack, software supply chain (SSC), code repositories, software vulnerabilities, cybersecurity framework, software bill of materials, artificial intelligence, machine learning, automation, penetration monitoring, defense industrial base, contractor software, software build security, third-party vendor security

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
U			UU	48	Vincent "Ted" Welsh
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**			**19b. TELEPHONE NUMBER** *(include area code)*
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED			443-360-4600

Standard Form 298 (Rev. 8/98)
Prescribed by ANSI Std. Z39.18

ON THE COVER:
(Source: Shutterstock & freepik)

THE AUTHOR

ABDUL RAHMAN, PH.D.

Dr. Abdul Rahman is a subject matter expert in the design and implementation of cloud analytics and architectures that support situational awareness tools for cybernetwork operations for commercial and government customers. He has over 25 years of information technology experience, including software development, network engineering, systems design, systems architecture, security, and network management. He has published widely on topics in physics, mathematics, and information technology. Dr. Rahman holds Doctor of Philosophy degrees in mathematics and physics.

ABSTRACT

The application of artificial intelligence (AI) to software supply chains (SSCs) within the defense industrial base (DIB) holds promise to improve cybersecurity posture, ensure stricter compliance with National Institute of Standards and Technology (NIST) controls, and increase user confidence in software built in part upon modules and libraries from outside repositories. AI can provide analysts with suggested frequencies for (re)scanning, supplement threat assessments of infrastructure, automate threat intelligence processing, and expedite cybersecurity risk management. Moreover, the security of SSCs in the DIB can benefit from similar uses of AI as a recommendation engine for communicating the probability of compromise. For U.S. Department of Defense cybersecurity analysts, AI-driven automation can provide insight into how closely software capabilities deployed on military and government networks adhere to NIST compliance standards. The ability to reflect the most up-to-date set of vulnerabilities within a system security plan could significantly improve upon the existing practice of relying on manual internal scanning. AI can enable human-in-the-loop workflows to optimize the integration of processed threat intelligence and better identify vulnerabilities per software and/or operating system. This report presents and discusses how AI can protect SSCs purpose-built for the DIB ecosystem.

ACKNOWLEDGMENTS

The author would like to thank the staff of the Cybersecurity & Information Systems Information Analysis Center and SURVICE Engineering Company for their guidance and review of this report.

EXECUTIVE SUMMARY

Managing the intricate and diverse supply chain within the U.S. government involves a heavy reliance on an extensive and varied network of suppliers and vendors for software components. This dependence introduces a range of challenges in ensuring the security of these software components. To address these software supply chain (SSC) security challenges effectively, a combination of technical solutions, robust security practices, collaboration among stakeholders, and adherence to industry standards is essential.

Prioritizing SSC security is critical for organizations to mitigate risks and safeguard against potential vulnerabilities and attacks. Unfortunately, federal entities often lack complete visibility into their SSCs, including information about the origin, integrity, and security of both packet and precursor components. This lack of visibility makes it challenging to identify and mitigate risks and vulnerabilities. Furthermore, reliance on third-party vendors introduces additional risks related to the security practices and integrity of provided software components.

To secure SSCs, it is crucial to implement preventive strategies against attacks. This can be achieved by establishing a security baseline and engaging in robust and continuous behavioral monitoring practices. The most sophisticated of these behavior-based methods involves the utilization of artificial intelligence (AI) models to forecast, infer, predict, correlate, and pinpoint likely weaknesses, potential attack vectors, and avenues of approach within SSC-embedded software. AI-powered systems can continuously monitor SSCs in real time, identifying suspicious activities and flagging actions that would otherwise allow for unauthorized access.

AI models are particularly well suited for the automation of routine SSC security audits and assessments that are intended to detect potential vulnerabilities, risks, and security control gaps. Such a proactive, real-time approach enables organizations to address potential exploits and vulnerabilities promptly and, if a penetration does occur, to receive immediate alerts to facilitate swift responses to security incidents, minimizing damage. Moreover, the integration of AI with security coding workflows can streamline the autocompletion and updating of required compliance practices, thereby enhancing overall code quality, defect reduction, and efficiency.

This Page Intentionally Left Blank

CONTENTS

CONTENTS, *continued*

SECTION 01 INTRODUCTION

Once used by the U.S. military in only its most high-tech systems, software is now omnipresent across the defense establishment. As the Defense Innovation Board noted in 2019, software drives "almost everything" that the U.S. Department of Defense (DoD) "operates and uses," from discrete weapons systems to the overarching networks that provide command, control, and communications capabilities for commanders [1]. While protecting DoD systems from traditional cyberbased attacks will remain an enduring challenge, threats to the security of the software supply chains (SSCs) that develop and produce critical products have recently risen in prominence as a preferred threat vector for penetrating and compromising information systems. By one estimate, the number of SSC attacks against commercial and public entities in the United States increased by more than 700% between 2019 and 2023 [2]. SSC attacks have become such an acute threat that the real-time tracking of SSC incidents has become a niche subsection of the cybersecurity solutions market [3].

1.1 DEFINING SSC ATTACKS

As its name suggests, an SSC refers both to the process of developing code-based packages across multiple parties and the outcome of chained-development activities into usable software products. SSCs encompass software modules, libraries, registries, and components, as well as all the hardware, operating systems, and cloud services that may be used during the coding and development process. As one leading software developer Red Hat has pointed out, an SSC is most properly considered to include even the people who write the code [4]. Current software development practices are relatively open, especially when compared with traditional coding methods, which remained in use well into the early 2000s. Instead of single entities developing software—entirely in house and by writing all code from scratch—current practices intentionally draw upon broad software communities. Developers leverage code sourced from external (but interconnected) libraries and modules that may serve different purposes for an application (e.g., encryption, authentication, and networking) [4].

Although this type of community development delivers key efficiencies to software production, it also presents bad actors with a wide range of potential threat vectors. Admitting dependencies through SSC development can introduce exploitable software code that is vulnerable to numerous, and cascading, vulnerabilities into the postbuilt product code baseline (see Figure 1-1). An SSC attack might seek to exploit open-source or shared tools, or to illicitly access a single developer's proprietary build infrastructures [5]. Whatever the vector, an SSC attack consists of at least two elements: (1) a malign actor compromising at least one supplier within an SSC and (2) that vulnerability then being used to harm other supplier(s) or the final product/customer. While it is possible that an SSC can be penetrated in part due to the actions of an insider, leading defense intelligence authorities like the U.S. National Counterintelligence and

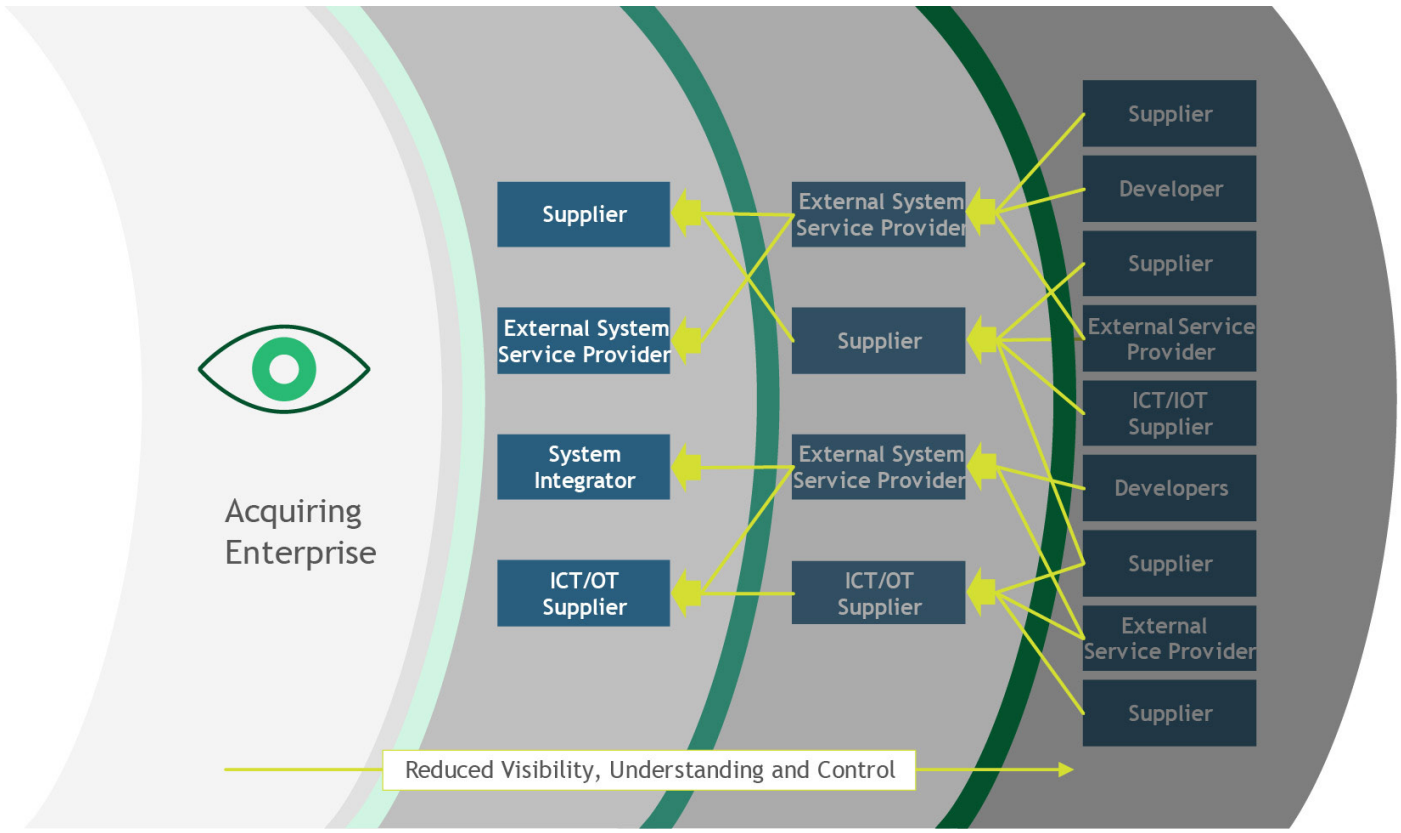

Figure 1-1. An Enterprise's Visibility, Understanding, and Control of Its SSC Decrease With Each Layer of the Broader Development Community's Involvement *(Source: Boyens et al. [6])*.

Security Center see cyberbased (or software enabled) SSC attacks as the more common and, thus, greater threat at present [5].

The documented ability to exploit vulnerabilities in an SSC has existed since at least the 1980s, when the "Ken Thompson hack" or "trusting trust attack" demonstrated the ability to compromise source code while leaving behind almost no trace of alteration [7]. Since then, the massive expansion of software production and the ubiquitous use of connected information systems across all sectors of the economy have made SSC exploits a prime vector for malign actors. For example, SSC attacks often target popular package managers (e.g., node package manager [npm] for Javascript node.js) and their user communities. These communities have experienced incredible growth over the past decade—the number of public repositories hosted in the GitHub platform grew from 46,000 in early 2009 to more than 200 million by 2022 [5]. Accordingly, adversarial nation-states, terrorists, and other transnational criminal organizations recognize that SSC attacks can cause widespread and cascading harmful effects, all while requiring relatively few resources to execute [8].

A number of headline penetrations in recent years have raised the profile of SSC attacks for malign actors. In 2017, the "NotPetya" SSC cyberattack—the most damaging such attack then to date—infected a line of accounting and tax reporting software used by the Ukrainian government before spreading to several large multinational firms. The malware that Russian-sponsored hackers inserted disrupted email systems at a major food manufacturer and disabled multiple logistics systems for an international shipping company. In doing so, NotPetya even crippled one pharmaceutical firm's ability to supply

vaccines to the U.S. Centers for Disease Control and Prevention [9]. By 2020, the "SolarWinds" cyberattack, which originated from the Russian Foreign Intelligence Service, similarly penetrated a wide array of networked systems, primarily within the U.S. federal government. After being injected with backdoor code, a routine software update package for a technology administration suite was widely downloaded; worse, the compromise went undetected for nearly 12 months [10].

1.2 SSCS AND THE DEFENSE INDUSTRIAL BASE

The DoD acquires software products and systems, professional services, and the supporting hardware and computing power needed for operation much in the same way it obtains crates of 5.56-mm rifle ammunition—mostly purchasing them from private firms and other public or nonprofit suppliers. Generally known as the Defense Industrial Base (DIB), this collection of organizations, facilities, and resources provides the DoD with hundreds of billions of dollars of products and services each year and represents the nation's enduring industrial and economic might [11]. The broad magnitude and scope of the DoD's acquisition activities means that more than 1 million workers and around 60,000 firms can be considered part of the DIB [11]. While many of these firms do not directly shape or influence the development of software products that enter militarily-relevant SSCs, every single entity (even those that only produce hardware, like 5.56-mm cartridges) uses software platforms that are vulnerable to penetration.

The DIB's immense scope and wide reach into suppliers and subcontractors make the defense of its SSCs an immense task. Two longstanding vulnerabilities further complicate this challenge:

1. The production of microelectronics, once common in the United States, has been mostly offshored to international producers, limiting government security oversight. (Enactment of the $54-billion federal "Creating Helpful Incentives to Produce Semiconductors (CHIPS) Act of 2022" is aimed at reversing this trend [12].)
2. "The growing complexity" of the electronics, platforms, and architectures that DIB-produced and DoD-operated systems depend upon makes SSC security an utterly overwhelming task. Both a "lack of traceability" and the need for persistent, "continuous monitoring" by the DoD of vendors and components in the DIB are key limiters in comprehensively securing SSCs within the national security and homeland defense space [13].

Along with the centrality of software to DoD operations, these two vulnerabilities have made penetration of SSCs within or adjacent to the DIB, as well as the intelligence community at large, a key objective for adversarial action [14]. In the past 5 years, military analysts have witnessed an uptick in attempts to penetrate defense-related SSCs, with a particular eye toward gaining direct control over DoD systems and other critical infrastructure to disable them in the event of armed conflict. In September 2019, hackers attacked the SSCs of four subcontractors working for Airbus, a major aeronautics firm that supplies the DoD with sensing systems as well as airframes [15]. In May 2023, a multi-agency joint advisory warned that a hacking group sponsored by the People's Republic of China, known as Volt Typhoon, had penetrated electrical systems in the homeland and in the U.S. territory of Guam—a key strategic site for operations in the U.S. Indo-Pacific Command [16]. Further complicating the daunting task of SSC security is the hodgepodge of systems, software vintages, and architectures that the DoD employs; each service branch largely operates its systems and networks separately from the others. Unifying a software security posture across the department has been likened to "assembling a puzzle with pieces from different sets" [17].

1.3 SECURING SSC

Both the DoD and the broader federal national security enterprise have responded to assess the vulnerability of their systems to SSC exploits and secure the broader software development and production communities that support government operations. For example, in July 2023, new administrative policies promulgated in the U.S. "National Cybersecurity Strategy Implementation Plan" tightened the technical requirements that suppliers and contractors must meet in following cybersecurity supply chain risk management (C-SCRM) best practices [18]. Operating in compliance with these best practices is a critical step to building trust in international software suppliers, as compliance makes the digital ecosystem more "transparent, secure, resilient, and trustworthy" [10].

Two months later, the DoD followed up the whole-of-government strategy with its own DoD-specific cyberstrategy [19]. The document recognizes, at a high strategic level, the importance of protecting the DIB from malicious cyberattacks and recommends a number of procedural changes, like the alignment of DIB contract incentives with DoD-specific cybersecurity requirements. Moreover, the strategy points toward the usefulness of ongoing research and development activities that might increase DoD capabilities for "rapid information-sharing and analysis" in the "identification, protection, detection, response, and recovery of critical DIB elements" [19]. The Office of the DoD Chief Information Officer is also working to finalize an enterprise-wide strategy for cyber supply chain risk management to guide protective actions for SSCs across the DoD [20].

The majority of technical guidance for securing SSCs across the firms and organizations that make up the DIB is generated by the National Institute for Standards and Technology (NIST). A longstanding federal entity originally involved in the standardization of weights, measures, and metrology measurements, NIST released its landmark cybersecurity framework (CSF) as Version 1.0 in 2014 [21]. The framework quickly found widespread adoption among commercial firms and government information technology (IT) departments and has been updated and expanded several times since [22].

At its core, the CSF details a set of best-practice cybersecurity activities, standardized tools, and references and further describes the "desired outcomes" of the application of the framework across an organization. While NIST is not a traditional regulatory agency, use of the CSF has since become mandatory for federal agencies [23]. Other NIST guidance, including the "Secure Software Development Framework (SSDF) Version 1.1: Recommendations for Mitigating the Risk of Software Vulnerabilities" (NIST Special Publication [SP] 800-218) [24] and the "Cybersecurity Supply Chain Risk Management Practices for Systems and Organizations" (NIST SP 800-161r1) [6], provides additional discussion of vulnerabilities and SSC security controls at both a technical and conceptual level (see Figure 1-2).

1.4 REPORT OVERVIEW

While guidance documents for the organizational practice of C-SCRM are very useful, they might also best be characterized as broad and nonspecific [25]. Moreover, as the volume of data and code that inhabit a given SSC continues to grow, entities like firms within the DIB would benefit greatly from next-generation analytical tools to identify potential SSC vulnerabilities and then secure them. Accordingly, this state-of-the-art report discusses the requirements, progress, and latest trends in using artificial intelligence (AI) tools and techniques to secure the defense-critical SSC. Detection of SSC attacks can be accomplished through building AI models deployed against collected distributed datasets designed, developed, trained, and tested over useful features. The combination of AI-enabled analytics with broader security

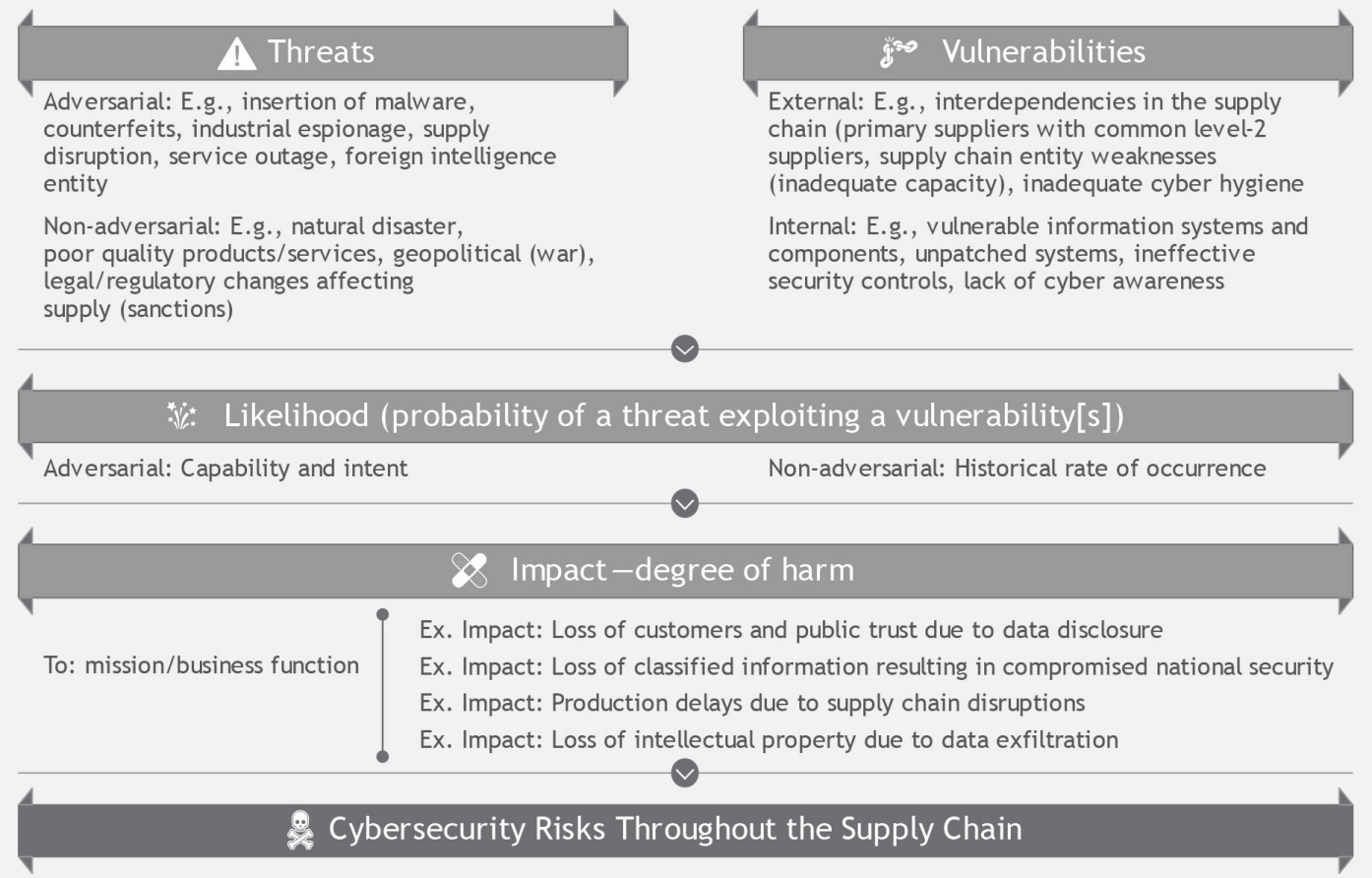

Figure 1-2. Cybersecurity Risks Throughout the Supply Chain *(Source: Boyens et al. [6])*.

approaches like the current version of NIST CSF 1.1 [26] (Version 2.0 of the CSF [27] is under development) can generate a truly comprehensive method of securing SSCs.

This report discusses data management strategies and feature development as the two core prerequisites for robust AI model development. Section 2 summarizes data management strategies to describe the most salient aspects needed for robust AI model development aligned to SSC security. Section 3, in surveying feature engineering and development, addresses the required understanding of SSC frameworks and their attributes upon which AI models will be trained. Section 4 explores how AI models can enhance software code reliability, integrate with blockchain technology, and improve SSC vulnerability analysis and detection. Overall, this report discusses the performance of AI models across all phases of SSC analytical processing, where it may lead to faster predictions and enhanced integration with security operations workflows.

This Page Intentionally Left Blank

SECTION 02 DATA MANAGEMENT STRATEGIES

The development of AI-enabled models is predicated first upon the use of robust and best-practice-compliant data management practices. The processes of data collection, aggregation, storage, and organization are key enablers of engineering (or developing) features targeted at the phenomena that will provide the largest benefits to early detection of SSC compromises. For instance, the use of packages from public component registries, if not carefully monitored, can introduce significant vulnerabilities to an SSC. Data provided by Sonatype, an SSC management company, reveal that the count of malicious packages identified across diverse open-source ecosystems in 2023 has tripled compared to the previous year [28]. That increase, in turn, comes on the heels of a staggering 650% year-on-year increase in security attacks exploiting vulnerabilities in open-source software's supply chain in 2021 [29].

2.1 OPEN-SOURCE PACKAGES

This rapid rate of expansion is truly remarkable, emphasizing the supply chain's emergence as one of the fastest-growing avenues for malevolent code execution. The widespread use of open-source packages in particular threatens to introduce vulnerabilities (or compromises) into a single SSC or multiple-linked, interdependent SSCs, with harmful ramifications that can cascade both upstream and downstream of a penetration [30] (see Figure 2-1). Without greater vision into the full reach of an SSC, benevolent actors are limited in the measures available to them to mitigate risk or employ countermeasures in a timely fashion.

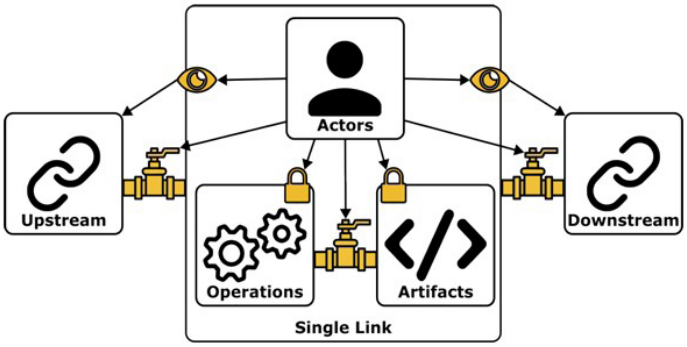

Figure 2-1. An SSC With Focus on a Single Link; Systemwide Security Depends on Upstream/Downstream Transparency, Link Validity, and Logical Separation Between Components and Links *(Source: Okafor et al. [30])*.

Virtually all modern software relies heavily on prior innovations distributed freely and made accessible by the world's most skilled experts. This invaluable foundation is offered to developers at no cost. As a result, it is often estimated that as much as 90% of the code utilized in software production systems is derived from open-source origins. However, a substantial number of open-source programming language repositories are maintained by the open-source community in a voluntary, part-time, and often haphazard manner [28]. While efforts have been made to prevent the hijacking of existing developer accounts for the dissemination of malicious components (such as the introduction of mandatory multifactor authentication), this does not fully deter attacks involving the upload of rogue packages from new accounts.

Few, if any, automated detection techniques are currently in place (much less actively used in practice), and the volunteer-based vulnerability removal procedures used by many community repositories are slow, cumbersome, and grossly inefficient when facing code intentionally designed to be malicious from the outset. Sonatype emphasizes the fact that packages harboring malicious code are often treated similarly to packages with new security vulnerabilities. This practice can allow malicious packages to persist longer than necessary, exposing developers to risks [28].

The potential of generative AI in software development is undeniably promising, but it does come with its set of challenges, both real and perceived. Significantly, a full 61% of the developers polled by Sonatype in 2023 view generic AI technology as "overhyped," while only 37% of IT security leads feel the same. While a majority of respondents currently utilize AI to varying degrees, that use is not always driven by personal preference. An astonishing 75% of both groups acknowledge feeling pressure from their organization's leadership to embrace and deploy AI technologies, as leadership typically stresses AI's productivity-enhancing capabilities over its associated potential security concerns [28]. However, it is likely that applying specific targeted AI models to the task of SSC vulnerability monitoring will minimize this skepticism, as AI moves from a nebulous technological concept to a series of discrete, defined, and useful software tools.

To proactively address the issue of open-source compromises, robust AI models can be implemented to support the prediction of package vulnerabilities that are susceptible to high-risk supply chain attacks. In 2022, Zahan et al. [29] focused on assisting software developers and security experts in assessing signals of weakness in the npm supply chain to prevent future attacks by conducting empirical investigations into npm package metadata. The authors scrutinized the metadata of 1.63 million packages, applying 6 indicators of compromise (IoC) of SSC security vulnerabilities. These include an expired maintainer domain, installation scripts, unmaintained packages, too many maintainers, too many contributors, and overloaded maintainers [29].

These IoCs can be used both to structure SSC data and formulate feature engineering approaches for AI models equipped to detect SSC attacks. One of the case studies used by the authors [29] identified more than 10 malicious packages using the installation script indicator. Furthermore, they discovered over 2,800 maintainer email addresses that were associated with expired domains—a vulnerability that could potentially enable an attacker to hijack over 8,000 packages by way of compromising npm accounts. The software development community provided positive feedback for the use of these IoCs as "weak link signals" or indicators. A survey completed by 470 npm package developers found greater than 50% support of responses for the use of 3 of the 6 IoCs: an expired maintainer domain, installation scripts, and unmaintained packages [29].

2.2 ATTACK SURFACE MANAGEMENT AND THREAT MODELING

Software package vulnerabilities are a significant contributor to the overall risk associated with software security. Eliminating all vulnerabilities is both impossible and impractical, as they can potentially lead to security risks in the SSC. Nevertheless, effective strategies exist for reducing and managing these risks. Two of the most effective strategies for managing supply chain security risks are known as "attack surface management" and "threat modeling."

The task of controlling attack surfaces involves assessing and managing the system entry points that attackers could exploit to compromise a system. Doing so helps to identify vulnerabilities

in either the system's design or implementation that might be particularly susceptible to malign action [31]. Threat modeling, on the other hand, is the process of analyzing and understanding the characteristics and scope of potential threats to a system—a key input to which is an assessment of its prime attack surfaces (see Figure 2-2) [31]. Both approaches are valuable to perform throughout the entire software lifecycle, from development and deployment to ongoing maintenance.

From an attack surface perspective, open-source code compromises transpire when malicious actors infiltrate publicly accessible code repositories and insert harmful code for public consumption. Unsuspecting developers—in their understandable search for freely available code snippets to fulfill specific functions—unwittingly incorporate these tainted elements into their third-party code.

One salient example dates back to 2018 and involved the detection of malevolent Python libraries on the official Python Package Index. Employing what is known as "typosquatting" tactics, the attacker fashioned libraries with names like "diango," "djago," and "dajngo," mimicking the common and much sought-after Python library correctly spelled as "django." To aid in the persistence of their propagation across linked SSCs, these deceptive libraries replicated the genuine code and functionality of their genuine counterpart but harbored additional features, such as the capability to establish boot persistence and create a reverse shell on remote workstations. Notably, open-source code compromises can also affect privately owned or enterprise software, since developers of proprietary code frequently incorporate open-source elements into their products [32]—sometimes even if their organization's security policy prohibits it.

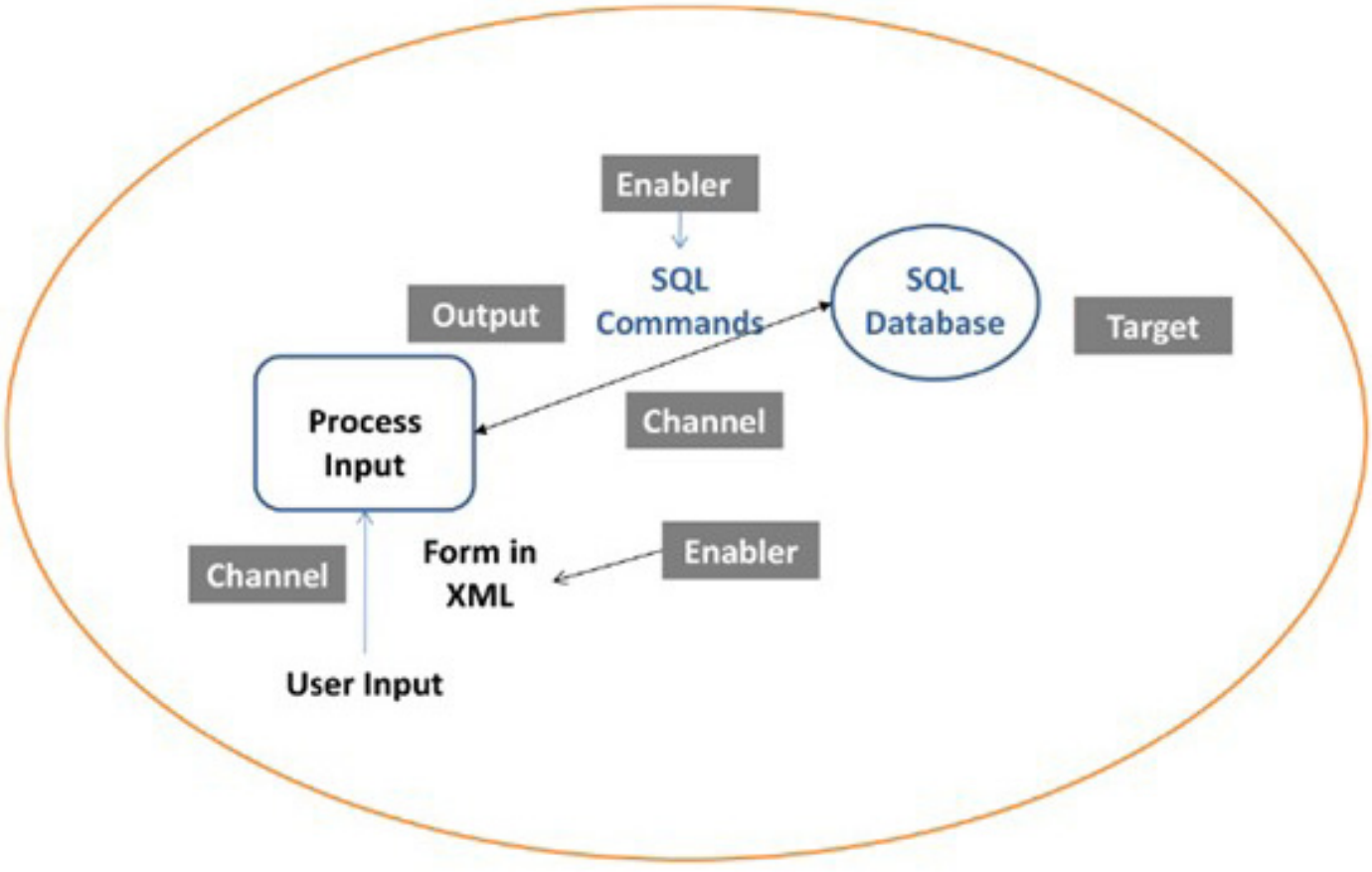

Figure 2-2. Data Flow Diagram of an Example Attack Surface *(Source: Ellison et al. [31])*.

Because an SSC's attack surface can admit a diverse and wide range of vulnerabilities, the ramifications of an SSC compromise can be dire. Initially, threat actors seek to exploit the "gaps" in a compromised software vendor to secure privileged and persistent access to a victim's network. By attacking an outside or third-party software vendor as part of their effort to target another organization, bad actors circumvent outer security measures like border routers and firewalls, thereby gaining an initial foothold. In the case of network access loss, threat actors can often simply re-enter the system through the compromised vendor.

While the process of gaining initial access is generally indiscriminate, threat actors often exercise discretion in selecting targets for subsequent actions. These follow-on actions exhibit considerable variability; however, they frequently commence with the insertion of tailored malware packages into a chosen target. Depending on the threat actor's intent and capabilities, this added malware may enable the attacker to conduct a variety of malicious activities, to include data or financial theft; surveillance of organizations or individuals; network or system disruption; or, in extreme cases, even physical harm or loss of life.

Those who work to defend friendly networks are limited when attempting to promptly mitigate the repercussions of an SSC compromise. This stems from the fact that organizations seldom have full control over their entire SSC, lacking the authority to compel each participant in the supply chain to swiftly undertake mitigation measures. Recognizing the challenge of mitigating postattack consequences, it is imperative for network defenders to proactively adopt and adhere to industry best practices. Implementation of these practices can only improve or enhance an organization's capacity to prevent, mitigate, and respond to such attacks.

Examining the attack surface and following known risk assessment methodologies (like threat modeling) are essential practices for mitigating SSC security risks. Nevertheless, it is crucial to acknowledge that these analyses are not static entities. Attackers have the capability—and are highly motivated—to introduce novel techniques that may infect software or code snippets that had previously been considered secure. Consequently, the assessment of the attack surface and its corresponding threat models should undergo periodic reviews via human-in-the-loop workflows and/or automated processes. The frequency of these reviews should be particularly heightened when dealing with emerging technologies (including the use of AI by third-party code-development processes elsewhere) and could align with the training, testing, and deployment of AI development lifecycles.

By their nature, new technologies may possess undocumented vulnerabilities (e.g., zero-days) because they lack an extensive history of known exploits, which would otherwise be used to inform threat modeling and other security risk assessment techniques. More frequent reviews are thus necessary to adapt to this evolving threat landscape. One such response might increase the frequency of internal system/enterprise scanning to detect abnormal behavior. In this use case, AI models can be developed and trained to specifically alert to such anomalies.

Both the frequent recalibration of the scope of security assessments and the behavior-based AI models can significantly aid the collection of essential information for organizational leaders to prioritize the means for their SSC security. Note that the security risks addressed by threat modeling and attack surface analyses differ significantly from those addressed by more traditional infrastructure security mechanisms, such as firewalls, authentication methods, and access control mechanisms. These infrastructure mechanisms primarily focus on preventing unauthorized access

to system resources. However, as systems continue to grow increasingly interconnected, and as security vulnerabilities in network and operating systems are reduced, application software is likely to emerge as the next most promising attack target for malign actors [31].

2.3 APPLICATION CODE SECURITY

The importance of application code security is often overlooked relative to SSC security, due to the assumption that standard network perimeter defenses (like network firewalls) can effectively block malicious access [31, 33–35]. However, application code itself can become a significant source of security risk, especially in complex software-use environments that have a wide and diverse user base, typically located across different organizations and partners. This potential weakness is exacerbated by the fact that many application-build personnel have not received adequate training in the practices of secure software development, making them unaware of the security risks of the design-and-coding choice made.

The oversight of application code security, however, is starting to be addressed. Numerous large-scale initiatives within organizations as diverse as banks, embedded systems manufacturers, software vendors, and military service branches like the U.S. Air Force are underway to enhance their overall software security via better application code management. These efforts follow a number of different cybersecurity frameworks, including SAFECode, the Building Security in Maturity Model (better known as BSIMM), the Software Assurance Processes and Practices Working Group, and the Software Assurance Maturity Model (known as SAMM) by the Open Worldwide Application Security Project (or OWASP) [34, 36]. Additionally, resources like the "build-security-in" website offer a host of valuable reference materials on software security practices for IT managers.

One noteworthy development is the increased use of fuzz testing, a technique that purposely uses malformed data to observe how applications respond to it. Unexpected application failures resulting from the ingest of malformed data can help to flag potential reliability and security issues. To date, fuzz testing has proven effective not only for security professionals but also for malign attackers themselves (for instance, in 2009, a fuzz testing tool was used by hackers to detect an exploitable defect in widely used extensible markup language [better known as XML] libraries).

2.4 NIST CYBERSECURITY FRAMEWORK

The inception of the NIST CSF can be traced back to Executive Order 13636, which was issued on 12 February 2013 [37]. Titled "Improving Critical Infrastructure Cybersecurity," the order marked the commencement of a number of endeavors on the federal level to facilitate the exchange of cybersecurity threat information and establish a comprehensive framework for mitigating cybersecurity risks. After CSF Version 1.0's release in 2014 [21], it was updated to Version 1.1 in 2018 [26]. At the time of writing, NIST had released a draft version of CSF 2.0 [27], which it intends to implement in the near future after first gathering and addressing feedback from key stakeholders and users [38]. Version 2.0 aims to be a "major update," as NIST describes it, expanding its coverage beyond critical infrastructure alone and broadening its scope to incorporate new technological developments and emerging issues like SSC security and ransomware [38].

The NIST CSF provides high-level guidance, developed through active engagement with, and valuable input from, stakeholders across government, industry, and academia. The CSF defines common terminology to promote homogeneity across organizations and harmonizes a number of pre-existing cybersecurity standards, guidelines, frameworks, and best practices into a systematic methodology to manage cybersecurity

risk [38]. CSF Version 1.1 classifies these management activities into one of five "framework functions:" (1) Identify, (2) Protect, (3) Detect, (4) Respond, and (5) Recover [26]. CSF Version 2.0 will add a sixth function: "Govern" (see Figure 2-3) [27]. Subordinate to the functions are numerous categories of actions (e.g., Protect > Awareness and Training) followed by subcategories, along with a number of informative references for each. In CSF Version 1.1, the "ID.SC" category represents activities for supply chain risk management under the identify function; at the subcategory level, this translates into five groups of SSC security activities (see Table 2-1). When collectively followed, these represent the essential (but notably, minimum) actions required to achieve effective C-SCRM [26].

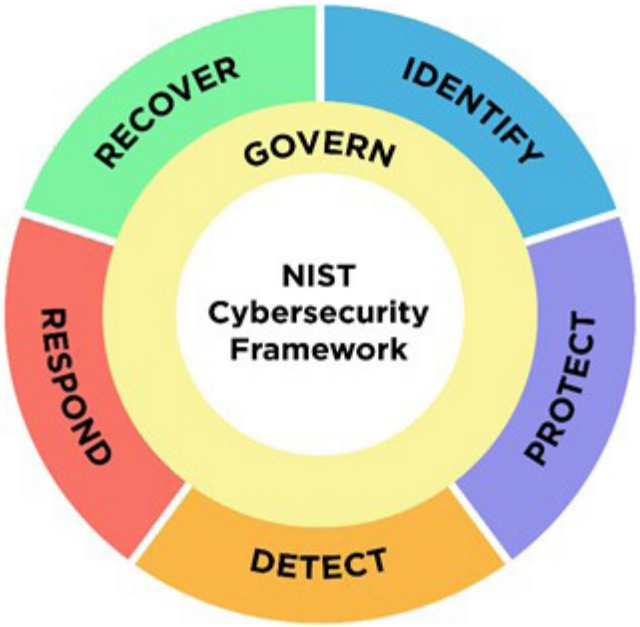

Figure 2-3. The Six Main Pillars of a Successful Cybersecurity Program, as Reflected in the NIST CSF Version 2.0 (Draft) *(Source: NIST [38])*.

Different aspects of a software product's distribution drive the requirements for SSC security readiness, but many are based on data management and/or cybersecurity best practices. Regarding data, the Data Management Body of Knowledge (known as DMBoK) can be used to transform an existing organization's perspective on how internal data assets are managed. (Internal assets can include but are not limited to the formalization of an accessible and usable data lake [loosely defined as a logical collection of data that is accessible but not overly structured] in addition to addressing the core requirements needed for SSC security.) To achieve this, current practices involve technology components like a data lake and are accompanied by data science "hooks" to enable model curation, development, and/or model development [39].

Despite the comprehensive protective nature of the CSF guidelines, it is evident that they alone are insufficient for some mission-critical SSCs, including many of those within the DIB. Rapid changes in the landscape of software security further underscore the need to go beyond perimeter defenses, threat modeling, and other traditional measures. SSC security demands the careful collection and continuous management of data sources so that AI models can build robust feature engineering methods into powerful AI models to detect SSC anomalies (i.e., those with strong performance metrics like accuracy, precision, and F1). AI-enabled tools will aid DIB entities to anticipate, infer, predict, correlate, and pinpoint potential vulnerabilities, potential intrusion routes, and attack vectors within software embedded in SSCs [39].

Table 2-1. NIST Guidance for Organizational Supply Chain Risk Management Under the "Identify" Function of the NIST CSF Version 1.1

Category	Subcategory	Informative References
Supply Chain Risk Management (ID.SC): The organization's priorities, constraints, risk tolerances, and assumptions are established and used to support risk decisions associated with managing supply chain risk. The organization has established and implemented the processes to identify, assess, and manage supply chain risks.	**ID.SC-1:** Cyber supply chain risk management processes are identified, established, assessed, managed, and agreed to by organizational stakeholders.	**CIS CSC 4** **COBIT 5** APO10.01, APO10.04, APO12.04, APO12.05, APO13.02, BAI01.03, BAI02.03, BAI04.02 **ISA 62443-2-1:2009** 4.3.4.2 **ISO/IEC 27001:2013** A.15.1.1, A.15.1.2, A.15.1.3, A.15.2.1, A.15.2.2 **NIST SP 800-53 Rev. 4** SA-9, SA-12, PM-9
	ID.SC-2: Suppliers and third-party partners of information systems, components, and services are identified, prioritized, and assessed using a cyber supply chain risk assessment process.	**COBIT 5** APO10.01, APO10.02, APO10.04, APO10.05, APO12.01, APO12.02, APO12.03, APO12.04, APO12.05, APO12.06, APO13.02, BAI02.03 **ISA 62443-2-1:2009** 4.2.3.1, 4.2.3.2, 4.2.3.3, 4.2.3.4, 4.2.3.6, 4.2.3.8, 4.2.3.9, 4.2.3.10, 4.2.3.12, 4.2.3.13, 4.2.3.14 **ISO/IEC 27001:2013** A.15.2.1, A.15.2.2 **NIST SP 800-53 Rev. 4** RA-2, RA-3, SA-12, SA-14, SA15, PM-9
	ID.SC-3: Contracts with suppliers and third-party partners are used to implement appropriate measures designed to meet the objectives of an organization's cybersecurity program and cyber supply chain risk management plan.	**COBIT 5** APO10.01, APO10.02, APO10.03, APO10.04, APO10.05 **ISA 62443-2-1:2009** 4.3.2.6.4, 4.3.2.6.7 **ISO/IEC 27001:2013** A.15.1.1, A.15.1.2, A.15.1.3 **NIST SP 800-53 Rev. 4** SA-9, SA-11, SA-12, PM-9
	ID.SC-4: Suppliers and third-party partners are routinely assessed using audits, test results, or other forms of evaluations to confirm they are meeting their contractual obligations.	**COBIT 5** APO10.01, APO10.03, APO10.04, APO10.05, MEA01.01, MEA01.02, MEA01.03, MEA01.04, MEA01.05 **ISA 62443-2-1:2009** 4.3.2.6.7 **ISA 62443-3-3:2013** SR 6.1 **ISO/IEC 27001:2013** A.15.2.1, A.15.2.2 **NIST SP 800-53 Rev. 4** AU-2, AU-6, AU-12, AU-16, PS-7, SA-9, SA-12
	ID.SC-5: Response and recovery planning and testing are conducted with suppliers and third-party providers.	**CIS CSC** 19, 20 **COBIT 5** DSS04.04 **ISA 62443-2-1:2009** 4.3.2.5.7, 4.3.4.5.11 **ISA 62443-3-3:2013** SR 2.8, SR 3.3, SR.6.1, SR 7.3, SR 7.4 **ISO/IEC 27001:2013** A.17.1.3 **NIST SP 800-53 Rev. 4** CP-2, CP-4, IR-3, IR-4, IR-6, IR8, IR9

Note: CIS = Center for Internet Security, CSC = Critical Security Control, COBIT = Control Objectives for Information and Related Technology, ISA = International Society of Automation, ISO = International Organization for Standardization, IEC = International Electrotechnical Commission. CIS CSC 4 [40], COBIT 5 [41], ISA 62443-2-1:2009 [42], ISO/IEC 27001:2013 [43], NIST SP 800-53 Revision 4 [44], ISA 62443-3-3:2013 [45], CIS CSC 19 [46], CIS CSC 20 [47]. As CSF Version 1.1 was released in 2018, some of the mentioned references may have been updated/revised. NIST maintains a live website that is consistently updated and can be used as a guide when searching for specific information [48].

This Page Intentionally Left Blank

SECTION 03 FEATURE DEVELOPMENT

In addition to Executive Order 14028 (12 May 2021) [49], which bolstered federal cybersecurity information-sharing requirements and imposed baseline security standards on government contractors, several other government initiatives and industry forums have recently addressed security assurance measures for bolstering the security of SSCs and all deployed software. One other NIST initiative, reflected in an August 2023 initial public draft titled "Strategies for the Integration of Software Supply Chain Security in DevSecOps CI/CD Pipelines" (NIST SP 800-204D, initial public draft), addresses multiple microservices, which are regarded as the "predominant" application architecture for cloud-native applications [39].

3.1 SECURE SOFTWARE UPDATES: DEVELOPMENT, SECURITY, AND OPERATIONS (DEVSECOPS); ARTIFICIAL INTELLIGENCE FOR INTERNET TECHNOLOCY OPERATIONS (AIOPS); AND MACHINE LEARNING OPERATIONS (MLOPS)

Cloud-native applications consist of multiple loosely connected components referred to as microservices. These software components follow an agile software development life cycle (SDLC) methodology known as DevSecOps, which employs continuous integration/continuous delivery (CI/CD) pipelines to streamline the development process. The security and integrity of these individual operations can significantly impact the overall security of an SSC, with threats potentially emerging from malicious actors' attack vectors or lapses in due diligence at nearly any point in the SDLC. Seamlessly integrating the diverse elements of SSC security assurance into CI/CD pipelines, as well as equipping organizations to effectively address SSC security during development and deployment, is central to defending cloud-native applications from penetration or SSC compromise [39].

NIST SP 800-204D makes it exceptionally clear that an essential component of any SSC is the software update process, one that is typically managed by specialized software development tools known as "software update systems" [39]. Maintaining the security and continuous monitoring of these update systems is absolutely paramount in ensuring the overall security of the SSC. The core function of a software update system is to identify the necessary files for a given update request and securely download those trusted files. Threats targeting software update systems focus predominantly on attempting to compromise the evidence generation process to cover their tracks, making it even more challenging to ascertain the legitimacy of updates.

Initially, it might seem that establishing trust in downloaded files only requires the execution of integrity and authenticity checks, conducted via the verification of file signatures and other associated metadata. However, the process of generating signatures itself is susceptible to known attacks (at minimum); this standing vulnerability

necessitates the imposition of additional security measures related to both signature generation and verification.

MLOps encompasses a set of methodologies that integrate the development and operational aspects of machine learning (ML) systems. The primary objective of MLOps is to optimize the entire lifecycle of ML, spanning from initial development and deployment to continuous monitoring and maintenance. In contrast to conventional software development (wherein development operations [known as DevOps] emphasize collaboration between developers and IT operations for automating and enhancing software delivery efficiency), MLOps extends these principles to the domain of ML and the powerful analytical tools that the ML space promises.

This extension acknowledges and addresses the distinctive challenges and prerequisites inherent to ML systems. The evolving security framework for integration of SSC security using DevSecOps CI/CD pipelines for software update systems laid out in NIST SP 800-204D has incorporated many of these essential security measures into its specifications and has recommended others for future implementations. This framework encompasses a collection of libraries, file formats, and utilities designed to enhance the security of both existing and new software update systems [39].

3.2 PUSH PROTECTION

A crucial security practice within an SSC during code commits involves preventing the inclusion of sensitive information within committed code. This safeguard is achieved through a scanning process designed to identify secrets, resulting in a feature known as "push protection." SSC security measures also extend to controls implemented during the continuous delivery process. For instance, one common control used during deployment is to verify whether the container image has been scanned for vulnerabilities and, if so, whether any vulnerabilities have been confirmed.

Following such an approach empowers DevSecOps teams to proactively maintain a secure container environment. It helps to ensure that only validated containers gain entry and assists overall in maintaining user trust during run time. This should also extend to the operational framework in support of AI model deployments typically relegated to AI practices. Implementing DevSecOps for AI (also called AIOps) enables streamlining and efficiency during the (re)training, (re)testing, and (re)deployment of AI models and is especially critical for those making behavior-based predictions of SSC threats. Furthermore, AIOps encourages a containerization approach, assuring that image deployment decisions align with organization-defined policies. Such alignment at run time must be achieved to prevent defects, vulnerabilities, and bad code from being introduced into models and/or software. These policies serve as the criteria for allowing or blocking the deployment of images, contributing to a robust security posture.

3.3 OTHER SSC FRAMEWORKS

Executive Order 14028 [49] (discussed previously) was prompted in part after a number of notable security breaches (including the SolarWinds hack discussed in Section 1.1) were met by a rising overall threat level posed by malicious cyberactors targeting software developers and contributors. The order entrusts various groups with the task of formulating new software security standards, tools, and best practices, and critically, it introduces a new category of "critical software," the precise definition of which has yet to be determined.

The order also removes certain barriers that hindered the sharing of cybersecurity threat information among government agencies. Beyond enhancing the cybersecurity of federal government systems, it urges private businesses and academia to elevate SSC security by adhering to the guidelines set forth by NIST. Furthermore, it has established a review board to evaluate and assess cybersecurity incidents and a playbook was

devised to govern the federal government's response to such incidents.

3.3.1 General Frameworks

Of particular significance to the software development field is the requirement for federal agencies to implement "zero trust" architectures, expedite the transition to secure cloud environments, and adopt additional data protection capabilities, along with related endpoint detection, response, and logging measures to mitigate ongoing supply chain risks. For businesses, the most notable aspect is the commitment to introducing new best practices and standards for SSCs. At the time of the order's issuance, it became evident that new compliance standards were on the horizon, though the exact form they would take was still unclear [49]. What has transpired is the evolution of a relatively recent concept known as a software bill of materials (SBOM), or an obligation to partake in vulnerability disclosure programs, and a requirement to demonstrate adherence to the best security practices.

The in-toto framework (intoto.io [50]) is a system designed to secure the entire SSC, encompassing the development, building, testing, and packaging processes. It provides attestation of integrity and verifiability for each action performed throughout the supply chain, including code writing, compilation, testing, and deployment. The framework ensures transparency by disclosing the order of steps and the actors involved. According to in-toto, the framework enables users to verify the intended execution of each step, authenticate the actors involved, and ensure that materials (such as source code) remain untampered between steps.

The Update Framework (TUF) empowers developers to safeguard update systems against repository compromises and attacks that focus on signing keys. TUF offers a robust approach to provide trust information about software, including meta-information about artifacts. Its primary objective is to authenticate the source of data stored in repositories. Additionally, TUF verifies the freshness of artifacts and maintains repository consistency, which are crucial steps for ensuring overall integrity and security in SSCs. TUF aims to prevent malicious behavior where attackers manipulate software artifacts in a way that the combined result becomes malicious [14].

The Open Software Supply Chain Attack Reference (OSC&R) provides objective insights into the target of an attack and its current phase. This perspective offers a holistic narrative that simplifies communication about security throughout an organization; delivers comprehensive visibility into coverage; and allows teams to assess potential impacts on the organization, evaluate the effectiveness of existing protective measures and controls, and prioritize responses.

3.3.2 SBOM and Pipeline Bill of Materials (PBOM)

While the concept of a bill of materials (BOM) outlining the components and their sources within a product is not novel, the novelty lies in devising a standard to apply to software and SSCs nationwide. Numerous manufacturing companies are already obligated to furnish a BOM detailing every component of a product, along with the original manufacturers in cases where they originate from third-party sources.

An application of BOMs familiar to the average consumer is for automobile vehicle recalls. In the event of a defective component, manufacturers can promptly identify the specific part and its source and determine how to rectify or replace it. This principle also applies to SBOMs, which offer a comprehensive inventory of all elements within the software, encompassing open-source libraries, third-party components, and proprietary code. This transparency assists organizations in managing their supply chains, proactively

identifying vulnerabilities, and responding more swiftly to security incidents. The use of SBOMs promises to ensure high-quality code, regulatory and security compliance, and protection against threats and vulnerabilities through the maintenance of an always-up-to-date and comprehensive BOM.

An SBOM functions similarly to a list of ingredients on food packaging, declaring the inventory of components used in constructing a software artifact, such as a software application. Just as individuals consult food labels to avoid allergenic ingredients, SBOMs aid organizations or individuals in steering clear of software that may pose risks [51]. In contrast to an SBOM, a PBOM goes beyond this inventory by informing users about whether other products produced in the same pipeline, using similar machinery, or within the same production facility contain potential vulnerabilities or risks [51]. PBOMs offer a comprehensive view, examining the entire pipeline from the design phase to production. This thorough assessment enhances the ability to avoid using harmful software by considering all the stages where a security breach could occur. PBOMs excel at helping users steer clear of potentially unsafe software because they scrutinize all stages where vulnerabilities or attacks might occur, as depicted in Figure 3-1.

These approaches provide guidelines, insights, communication, and declarations for trackable and referenceable artifacts as inputs in support of security of SSCs from one or many of these sources to determine possible anomalies in source, build, availability, and/or distribution of software.

3.3.3 Supply Chain Levels for Software Artifacts (SLSA)

SLSA is a framework designed to categorize various software artifacts within a supply chain, based on their integrity level [52]. In the context of SLSA, "integrity" signifies the confidence that a software artifact has not been tampered with or altered in an unauthorized manner, ensuring it remains in its original and intended state. On the other hand, the OSC&R framework offers a systematic

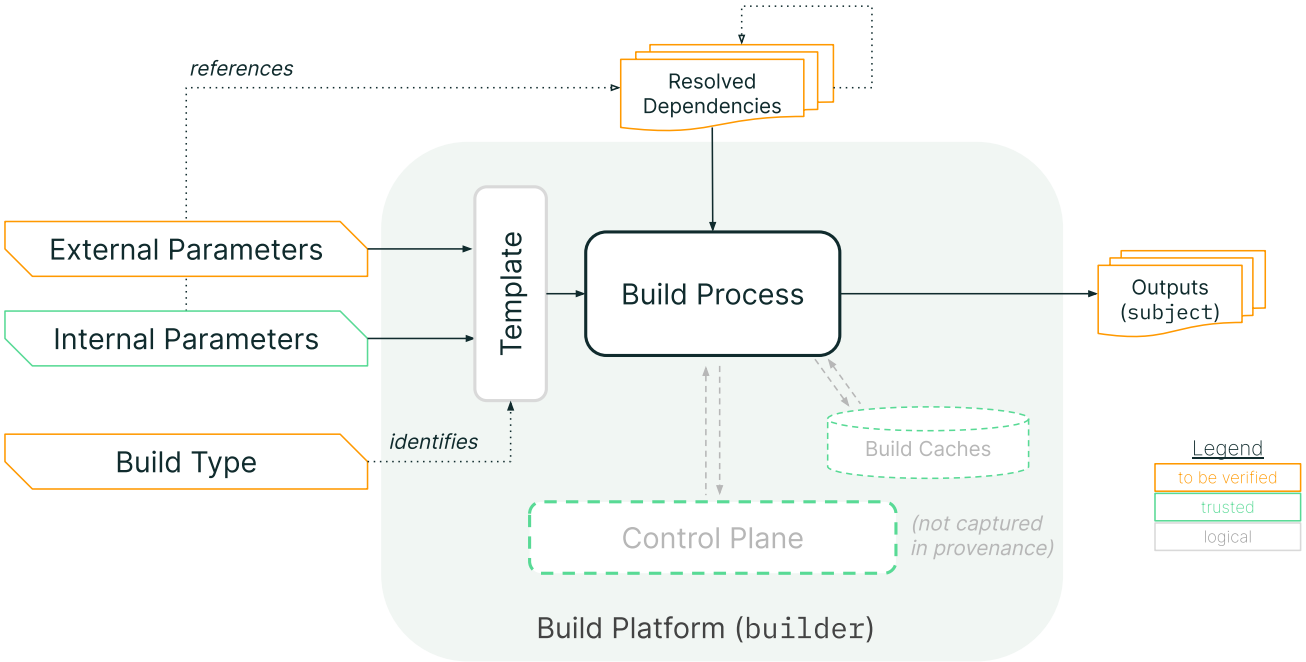

Figure 3-1. Build Platform Workflow for Provenance, as Attestation of Created Artifacts in Support of SSC Security *(Source: The Linux Foundation [52])*.

and actionable approach to comprehending attacker behaviors and techniques employed in compromising the SSC [50].

Supply chain threats take on different characteristics that are incorporated into mitigations with SLSA Version 1.0 [52]. The four threats are: (1) sources, (2) dependencies, (3) builds, and (4) availability/verification.

Source integrity threats involve the potential for an adversary to tamper with the source code of a software product, making unauthorized changes [52]. They can also encompass insider threats, where authorized individuals introduce unauthorized changes. Dependency threats involve adversaries introducing malicious behavior into a software artifact by targeting its external dependencies. SLSA helps mitigate these threats when one verifies the provenance (origin and authenticity) of dependencies using the SLSA framework. Build threats pertain to adversaries potentially introducing malicious behavior into a software artifact without changing its source code. They also include situations where artifacts are built from unintended sources or dependencies. The SLSA build track can help mitigate these threats by allowing consumers to verify that received artifacts were built as expected (see Figure 3-2). Finally, availability threats involve adversaries attempting to deny access to source code or the ability to build a package, effectively disrupting the availability of software resources [52].

The SLSA framework outlines specifications for distributing provenance information and further defines the connection between build artifacts and their associated provenance (known

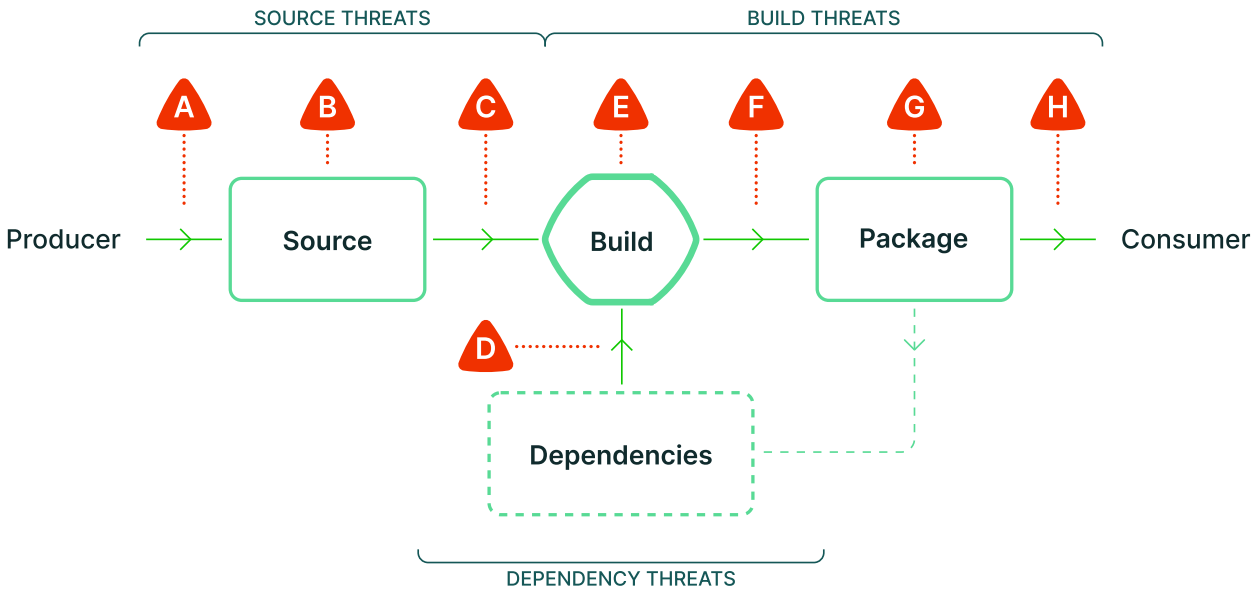

Figure 3-2. SLSA Approach to SSC Threats and Mitigations *(Source: The Linux Foundation [52])*.

as build attestations) [38]. Its primary focus is on ecosystems that distribute build artifacts, although it also extends its attention to ecosystems distributing container images or solely providing source artifacts. Many of the core principles within SLSA are applicable to various types of artifacts. To ensure that provenance information remains accessible for verification after artifact generation, SLSA mandates the distribution and verification of provenance metadata in the form of SLSA attestations.

Within this framework, it is the responsibility of the package ecosystem to make its expectations available to consumers, reliably redistribute both artifacts and their associated provenance, and furnish the necessary tools for secure artifact consumption. The "package ecosystem" denotes a set of rules and conventions governing the distribution of packages. It is important to note that every package artifact belongs to an ecosystem, whether formal or informal. Many ecosystems are informal, while some ecosystems are formally recognized, such as those governing language (e.g., Python/Python Packaging Authority), operating systems (e.g., Debian/ Advanced Package Tool), or general artifact distribution (e.g., Open Container Initiative [known as OCI]).

Conversely, informal ecosystems can exist within organizations or even within ad hoc distribution methods like sharing software through a website link, all of which are considered "ecosystems" within the context of SLSA. During the package upload process, a package ecosystem has the option to verify that the artifact's provenance aligns with the expected values for that package's provenance before accepting it into the package registry. This practice is strongly recommended whenever feasible, as it benefits all consumers within the package ecosystem.

Furthermore, SLSA offers valuable insights for artifact distributors on how to incorporate the distribution of SLSA provenance effectively. Its primary concern revolves around the methods of distributing attestations and establishing the relationship between attestations and build artifacts, rather than prescribing a specific format for attestations themselves. One noteworthy aspect of SLSA is that it encourages attestations to be bound to individual artifacts rather than releases. This approach acknowledges that a single "release" of a project, package, or library may encompass multiple artifacts, which in turn come from builds on multiple different platforms, architectures, or environments. These builds may not necessarily occur simultaneously and can even span multiple days.

In many ecosystems, determining when a release is considered "complete" can be an exceptionally challenging task. It is often permissible to add new artifacts to older releases during the normal process of adding support for new platforms or architectures. As a result, the collection of attestations for a given release is likely to expand substantially over time, as additional builds and attestations are created and accrued. Therefore, package ecosystems are advised to support multiple individual attestations per release, allowing the relevant provenance for each build to be associated with the release as needed, depending on its relationship to the associated artifacts.

SECTION 04 APPLICATIONS OF AI

To enhance SSC security, it is imperative to turn to AI-enabled models and security constructs that significantly improve upon the semi-automated (and sometimes manual) processes that are currently used to forecast, infer, predict, correlate, and identify likely weaknesses, potential attack vectors, and avenues of approach within SSC-embedded software. Promising areas within this task include the integration of blockchain technology with SSC cybersecurity frameworks, the use of AI-enabled models to better automate software vulnerability analysis, and the use of AI to enhance the guarantee and confirmation of code reliability.

4.1 AI MODELS WITH BLOCKCHAIN INTEGRATION WITH SSC FRAMEWORKS

One line of development meriting special discussion is the use of blockchain technology and methods for enhanced SSC security. Blockchain technology offers an encrypted, peer-to-peer digital ledger accessible to network participants, whether public or private. It establishes a decentralized trust system without the need for trusted third parties. Within a blockchain network, numerous partners or nodes coexist, with each node possessing a copy of the maintained data. The data within the blockchain are structured into blocks, where each block comprises a collection of records, commonly referred to as transactions. These transactions are organized into a Merkle tree, wherein the transaction records serve as the leaves and each child node hash acts as an intermediary node.

Smart contracts function as trusted intermediaries positioned between blockchain clients and blockchain storage, enabling advanced functionalities. They facilitate client requests, where the logic for processing services, ranging from simple to complex, can encompass tasks like validating application state, enforcing governance rules, or conducting credential checks. Smart contracts streamline interactions with the underlying blockchain by executing queries to store or retrieve data through a programmable interface [53–55].

Motivated from research [50, 52–55], an integrated SSC-blockchain platform (e.g., Let'sTrace) can be specifically designed to manage software patch releases and ensure the integrity of these patches through the utilization of blockchain-based smart contracts. The verification of patch integrity for software products is conducted using an SSC framework (e.g., TUF), and this verification functionality is seamlessly integrated into the smart contracts within the platform. Before a patch is deployed, it can undergo verification through the platform utilities. Additionally, all software updates must adhere to the SSC protocols (e.g., in-toto) and include relevant metadata files.

When a vendor releases a patch, it undergoes rigorous testing and a comprehensive summary is then uploaded to the platform. This summary includes details such as the patch updates, affected software modules, and alterations in network traffic patterns (both incoming and outgoing) following the patch update. Participating utilities

in the platform have the capability to view and verify these patch updates and patch supply chain information using in-toto. The platform can also construct AI and ML models utilizing supply chain data from various peers, employing a federated learning (FL) system. These models can subsequently be integrated into blockchain smart contracts to enhance the verification of supply chain data (see Figure 4-1).

In the event that any suspicious incidents related to a software or patch update are detected

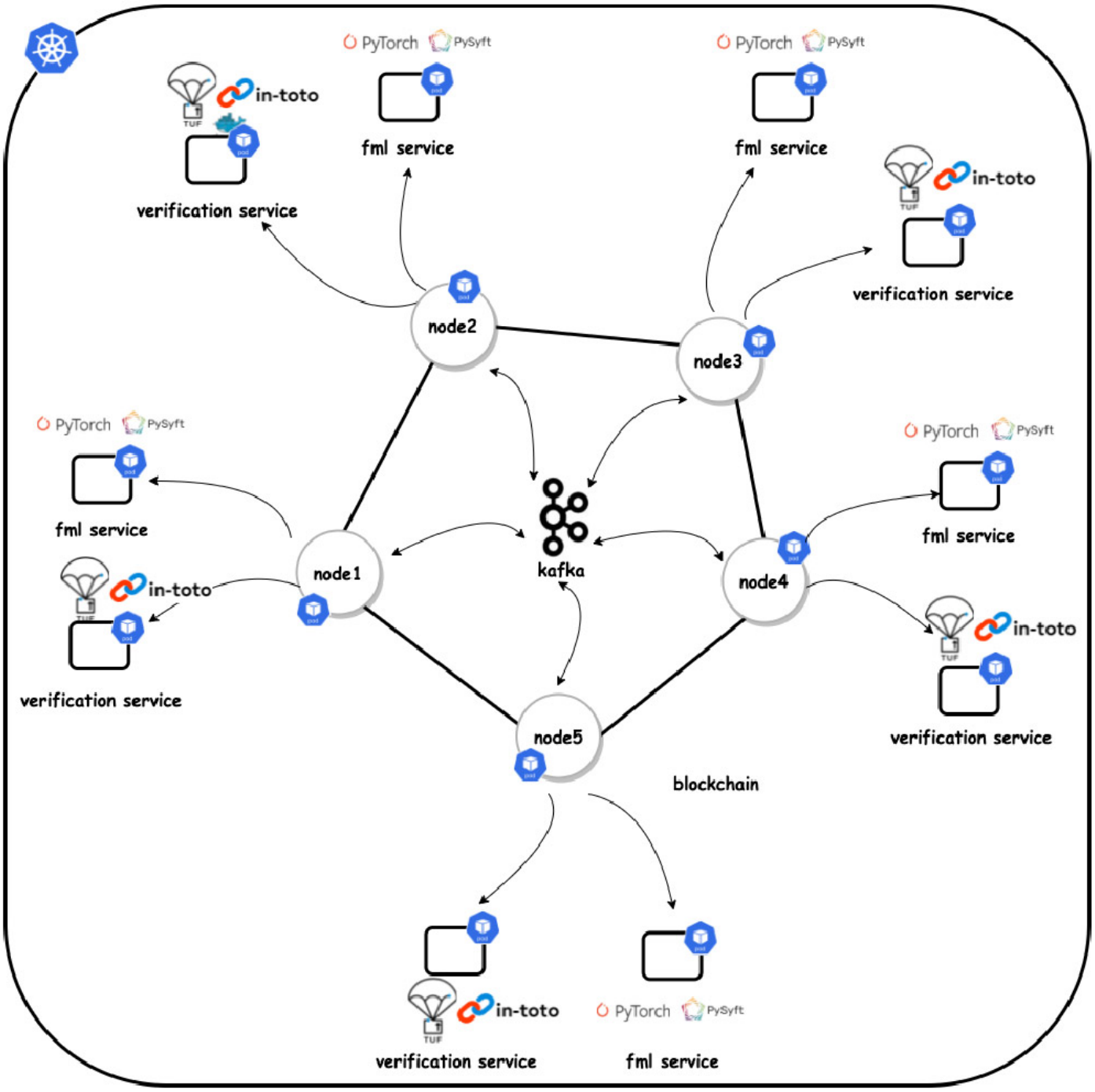

Figure 4-1. Notional Architecture of Blockchain Integrated With AI (FL) and Framework; Frameworks Provide Artifact Level Alignment for Distributed AI (FL) to Be Trained Over All Locations *(Source: Bandara et al. [55])*.

during the verification process, immediate reporting and notification to utilities and vendors within the platform are facilitated. This enables prompt actions to be taken to mitigate potential vulnerabilities. The ability to offer additional fidelity using annotations from PBOMs and SBOMs can also be used for training/testing these models for detection of more nuanced SSC compromises.

The immutability of blockchain provides the ability to perform change detection on a "true" state of the system that could not be altered by hackers. This offers the opportunity to employ a variety of continuous monitoring tools coupled with AI models to detect changes, while also storing alterations to references, pointers, and values relevant for SSC artifacts. These artifacts could be driven from PBOM artifact and SBOM artifact annotations aligned with NIST SP 800-204D [39, 51].

In light of the escalating demand for application software and the industry's race for swift code development, an enduring challenge remains in maintaining both speed and the production of bug-free software, particularly in the context of the now-common postpandemic, work-from-home setup where constant supervision of software developers is not always assured. This scenario increases the likelihood of introducing software bugs, and traditional testing methods are likely to struggle in delivering optimal performance.

To address this challenge, innovative decentralized software testing systems that leverage AI and/or blockchain technologies automatically detect and prevent the injection of vulnerable code by combining the capabilities of deep learning with the power of smart-contract-driven blockchain. This approach eliminates the reliance on manually written rules for vulnerability detection. The range of nonvulnerability scoring is broad enough that a discrete score effectively communicates the classification of the source code. Additionally, integration of an InterPlanetary File System can ensure efficient storage within the blockchain [56].

4.2 SOFTWARE VULNERABILITY ANALYSIS AND DETECTION USING AI

SSC vulnerabilities deriving from software typically arise from design flaws or implementation errors, posing threats to the security of a system. At present, the most prevalent approach for identifying such vulnerabilities is known as static analysis. Many existing technologies in this domain rely on rules or code similarity at the source code level, using a manually defined matching process to identify vulnerability features. However, accurately defining and designing these rules and features is a significant challenge, with high labor inputs, lack of speed, and other considerations limiting the practical application of static analysis beyond single use cases.

To address this issue, some researchers have advocated for the use of neural networks, sporting automatic feature extraction capabilities, to enhance the intelligence of detection. Yet, a wide variety in different types of neural networks—and the substantial impact of different data preprocessing methods on model performance—presents formidable challenges for engineers and researchers who would aim to select appropriate, much less optimal combinations to resolve a given problem.

Recently, researchers have conducted extensive experiments in this space. Those that have produced the most promising results focused on the two most common neural networks (bidirectional long short-term memory [Bi-LSTM] and random vector functional link [RVFL]) and the two most classical data-preprocessing methods (vector representation and program symbolization) in the context of software vulnerability detection. Their findings offer valuable insights:

- RVFL consistently exhibits faster training speed than Bi-LSTM, while the latter boasts higher prediction accuracy

- Utilizing doc2vec for vector representation enhances training speed and generalization ability compared to word2vec
- Multilevel symbolization proves beneficial in enhancing the precision of neural network models

These lessons can serve as practical guidelines for both researchers and engineers in navigating the complexities of neural network selection and data preprocessing for software vulnerability detection [57].

The growing complexity of software applications and the imperative to minimize vulnerabilities have already spurred the creation (and limited adoption) of ML techniques for identifying software vulnerabilities in source code. However, many of these existing lack the accuracy required for ready use in an industrial context. One recent study [57] introduces a novel approach, utilizing an abstract syntax tree neural network (ASTNN), to identify and classify software vulnerabilities according to common weakness enumeration (CWE) types. The study proceedings follow two key assertions: (1) ASTNN outperforms previous ML neural network architectures, and (2) the benchmark dataset commonly used for ML vulnerability classification is inadequate for this purpose. The ASTNN architecture is detailed, and an evaluation using over 44,000 test cases across 29 CWEs in the NIST Juliet Test Suite dataset yielded a minimum accuracy of 88% across all CWEs [58].

4.3 AI-ENHANCED CODING RELIABILITY

Especially since 2022, when large language model-enabled AI "chatbot" interfaces began receiving an immense amount of interest and discussion in the AI space, many commercial engineers have looked to apply AI directly to coding practice. Doing so has transformed the approach to coding that many take, bringing about increased efficiencies in static analysis, reliability, and defect reduction. The following examples highlight current projects available for download and implementation; note that each is likely to evolve in a dynamic fashion, given the current, nascent stage of AI-enabled code enhancement.

GitHub, a leading code hosting platform, has released (and also uses) CodeQL, an AI-powered static analysis tool for discovering vulnerabilities across a codebase. CodeQL employs sophisticated AI algorithms to automatically identify security vulnerabilities in code. Beyond detection, it also suggests fixes and patches, proactively addressing security concerns for millions of open-source projects hosted on GitHub. CodeQL contributes significantly to the overall security of the software ecosystem by identifying and addressing vulnerabilities early in the development process. Using CodeQL to identify a vulnerability promises to eradicate it and all possible variants "forever" [59].

Facebook (Meta Platforms, Inc.) has developed Infer, an AI-based code analysis tool, to enhance software reliability and prevent issues from reaching production codebases. Used by a wide array of both SSC-centric and other commercial entities (including Amazon Web Services, Uber, Microsoft, and Sonatype), Infer utilizes static analysis to identify various programming errors and potential crashes, even in complex and large-scale codebases. By catching bugs before they propagate, Infer helps to maintain high-quality and stable applications, reducing postrelease bug fixes. In addition to marketing the Infer service externally, Facebook runs it continuously within every code modification for its main user applications, including Facebook itself (Android/iOS), Facebook Messenger, Instagram, and others [60].

Developed by Google (Alphabet, Inc.), DeepCode AI is an AI-driven code review tool that extends beyond error detection. It provides intelligent suggestions for code improvements, offering specific recommendations to developers. DeepCode AI analyzes code patterns, individual

coding styles, and best practices to assist developers in writing cleaner, more efficient code. This can not only reduce the likelihood of errors but can accelerate software package development overall by automating code enhancements. Built upon multiple AI models, DeepCode AI has also been trained specifically on open-source code-security data. DeepCode is particularly valuable for optimizing development workflows and reducing coding errors, ultimately saving time and resources [61].

IntelliCode, developed by Microsoft and marketed within its Visual Studio integrated development environment, enhances the code review process by offering AI-generated code completion suggestions and recommendations. IntelliCode achieves this via continual analysis of coding patterns, contextual information, code type, and more; it then generates recommendations on its assessment of GitHub open-source code contributions. For instance, IntelliCode detects repetition in a programmer's draft code, including the use of variable (or near-match) names; this alone removes a common vulnerability, as attackers can exploit duplicate code in some scenarios to gain access [62].

This Page Intentionally Left Blank

SECTION 05 CONCLUSIONS

As AI-enabled tools of all types continue to find increasingly rapid purchase in both private industry and government, their application to cybersecurity practices in general, and SSC defense in particular, are almost certain to follow suit. Overall, one of AI's most useful capabilities is how well it can filter through large volumes of data and detect low signal-to-noise advanced-persistent-threat-like risks. Markedly better detection will enable faster prediction speeds and, ultimately, rapid-response actions. Future work in the AI-enabled SSC space is likely to investigate the ability to accelerate behavior-based detections via increased compute, optimal AIOps, integrated end-to-end workflow, and dynamic updates for feature engineering. With the potential addition of automated distributed training, future tools could easily display the ability to detect malign cyberactions, based on behavioral data, while using AI to support near-real-time alerting.

Effectively managing the intricate and diverse supply chain within the U.S. government entails a heavy reliance on an extensive network of suppliers and vendors providing software components. This dependence poses challenges in ensuring the security of these components within the SSC. To address these security challenges comprehensively, a combination of technical solutions, robust security practices, collaborative efforts among stakeholders, and adherence to industry standards is imperative.

Prioritizing security within the SSC is crucial for organizations to mitigate risks and protect against potential vulnerabilities and attacks. Unfortunately, federal entities often lack complete visibility into their SSCs, including details about the origin, integrity, and security of the components. This limited visibility makes it difficult to identify and address potential risks and vulnerabilities. Moreover, relying on third-party vendors introduces additional risks related to the security practices and integrity of the provided software components.

To enhance SSC security, it is essential to implement preventive strategies against potential attacks. This involves establishing a security baseline and adopting robust behavioral continuous monitoring practices. Behavioral methods leverage AI models to forecast, infer, predict, correlate, and identify likely weaknesses, potential attack vectors, and avenues of approach within SSC-embedded software. AI-powered systems can monitor SSCs in real time, detecting suspicious activities and unauthorized access.

AI is particularly adept at automating routine security audits and assessments of SSCs to identify potential vulnerabilities, risks, and security control gaps. This proactive approach enables organizations to address potential exploits and vulnerabilities promptly, receiving timely alerts for swift responses to security incidents and minimizing potential damage. Furthermore, integrating AI with security coding workflows

streamlines the autocompletion and updating of required compliance practices, enhancing overall code quality, reducing defects, and improving efficiency.

REFERENCES

1. Defense Innovation Board. "Software Acquisition and Practices (SWAP) Study, Main Report." U.S. Department of Defense, Arlington, VA, https://media.defense.gov/2019/May/01/2002126693/-1/-1/0/SWAP%20MAIN%20REPORT.PDF, 3 May 2019.
2. Hughes, C. "NIST Provides Solid Guidance on Software Supply Chain Security in DevSecOps." *CSO Online*, https://www.csoonline.com/article/655648/nist-provides-solid-guidance-on-software-supply-chain-security-in-devsecops.html, 19 October 2023.
3. Comparitech Limited. "Worldwide Software Supply Chain Attacks Tracker (Updated Daily)." *Compairtech*, https://www.comparitech.com/software-supply-chain-attacks/, 29 September 2023.
4. Red Hat, Inc. "What Is Software Supply Chain Security?" *Red Hat*, https://www.redhat.com/en/topics/security/what-is-software-supply-chain-security, 14 December 2022.
5. Director of National Intelligence. "Software Supply Chain Attacks." National Counterintelligence and Security Center, Washington, DC, https://www.dni.gov/files/NCSC/documents/supplychain/Software-Supply-Chain-Attacks.pdf, 21 April 2023.
6. Boyens, J., A. Smith, N. Bartol, K. Winkler, A. Holbrook, and M. Fallon. "Cybersecurity Supply Chain Risk Management Practices for Systems and Organizations." NIST SP 800-161r1, National Institute for Standards and Technology, Gaithersburg, MD, https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-161r1.pdf, May 2022.
7. Thompson, K. "Reflections on Trusting Trust." *Communications of the ACM*, vol. 27, no. 8, https://dl.acm.org/doi/pdf/10.1145/358198.358210, August 1984.
8. GitHub, Inc. "Catalog of Supply Chain Compromises." *Cloud Native Computing Foundation*, https://github.com/cncf/tag-security/tree/main/supply-chain-security/compromises#catalog-of-supply-chain-compromises, June 2023.
9. Crosignani, M., M. Macchiavelli, and A. F. Silva. "Pirates Without Borders: The Propagation of Cyberattacks Through Firms' Supply Chains." Staff Report No. 937, Federal Reserve Bank of New York, New York, NY, https://www.newyorkfed.org/medialibrary/media/research/staff_reports/sr937.pdf, July 2021.
10. U.S. Government Accountability Office. "Cybersecurity: Federal Response to SolarWinds and Microsoft Exchange Incidents." GAO-22-104746, report to congressional addressees, Washington, DC, https://www.gao.gov/products/gao-22-104746, 13 January 2022.
11. Nicastro, L. A. "The U.S. Defense Industrial Base: Background and Issues for Congress." R47751, Congressional Research Service, Washington, DC, https://s3.documentcloud.org/documents/24039377/r47751.pdf, 12 October 2023.
12. 117th Congress. "Creating Helpful Incentives to Produce Semiconductors (CHIPS) Act of 2022." Pub. L. No. 117–167, 136 Stat. 1366, https://www.govinfo.gov/content/pkg/PLAW-117publ167/pdf/PLAW-117publ167.pdf, 9 August 2022.
13. Ebert, M. "Attacking the Cyber Supply Chain Problem at Its Source—Shifting Way Left!" U.S. Army Redstone Test Center, presentation before the National Cyber Summit, Huntsville, AL, https://eventpower-res.cloudinary.com/files/v1/media/National%20Cyber%20Security%20S/23ncs/presentation_files/Attacking%20the%20Cyber%20Suppl/ozpdhpkmsmrgc6nt5o9q.pdf/Attacking_the_Cyber_Supply_Chai_Dr_Jacob_Cox_Jr_NCS_2023_C-SCRM_Final, 20 September 2023.
14. Thompson, L. "Five Reasons Software Is Eclipsing Hardware in Pentagon Technology Plans." *Forbes*, https://www.forbes.com/sites/lorenthompson/2023/08/14/five-reasons-software-is-eclipsing-hardware-in-pentagon-technology-plans/?sh=2ba6bd8d6c7e, 14 August 2023.
15. Center for Strategic & International Studies. "Significant Cyber Incidents." *CSIS*, https://www.csis.org/programs/strategic-technologies-program/significant-cyber-incidents, October 2023.
16. Greenberg, A. "China-Linked Hackers Breached a Power Grid—Again." *Wired*, https://www.wired.com/story/china-redfly-power-grid-cyberattack-asia/, 12 September 2023.
17. Pavithran, A. "The Pentagon Is Running out of Time to Get Zero Trust Right." *C4ISRnet*, https://www.c4isrnet.com/opinion/2023/10/17/the-pentagon-is-running-out-of-time-to-get-zero-trust-right/, 17 October 2023.
18. The White House. "National Cybersecurity Strategy Implementation Plan." Washington, DC, https://www.whitehouse.gov/wp-content/uploads/2023/07/National-Cybersecurity-Strategy-Implementation-Plan-WH.gov_.pdf, 13 July 2023.
19. U.S. Department of Defense. "Summary: 2023 Cyber Strategy of the Department of Defense." Arlington, VA, https://media.defense.gov/2023/Sep/12/2003299076/

REFERENCES, *continued*

-1/-1/1/2023_DOD_Cyber_Strategy_Summary.PDF, September 2023.

20. U.S. Government Accountability Office. "Information and Communications Technology: DoD Needs to Fully Implement Foundational Practices to Manage Supply Chain Risks." GAO-23-105612, report to congressional committees, Washington, DC, https://www.gao.gov/assets/gao-23-105612.pdf, 18 May 2023.
21. National Institute for Standards and Technology. "Framework for Improving Critical Infrastructure Cybersecurity, Version 1.0." Gaithersburg, MD, https://www.nist.gov/system/files/documents/cyberframework/cybersecurity-framework-021214.pdf, 12 February 2014.
22. Otto, G. "NIST Wants More Feedback on Cybersecurity Framework." *FedScoop*, https://fedscoop.com/nist-looking-for-addtional-feedback-on-cybersecurity-framework/, 10 December 2015.
23. U.S. Executive Office of the President. "Strengthening the Cybersecurity of Federal Networks and Critical Infrastructure." *National Archives*, Executive Order 13800, 82 FR 22391, https://www.federalregister.gov/documents/2017/05/16/2017-10004/strengthening-the-cybersecurity-of-federal-networks-and-critical-infrastructure, 11 May 2017.
24. Souppaya, M., K. Scarfone, and D. Dodson. "Secure Software Development Framework (SSDF) Version 1.1: Recommendations for Mitigating the Risk of Software Vulnerabilities." NIST SP 800-218, National Institute for Standards and Technology, Gaithersburg, MD, https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-218.pdf, February 2022.
25. Boyens, J., C. Paulsen, N. Bartol, K. Winkler, and J. Gimbi. "Key Practices in Cyber Supply Chain Risk Management: Observations From Industry." NISTIR 8276, National Institute for Standards and Technology, Gaithersburg, MD, https://nvlpubs.nist.gov/nistpubs/ir/2021/NIST.IR.8276.pdf, February 2021.
26. National Institute for Standards and Technology. "Framework for Improving Critical Infrastructure Cybersecurity, Version 1.1." Gaithersburg, MD, https://nvlpubs.nist.gov/nistpubs/CSWP/NIST.CSWP.04162018.pdf, 16 April 2018.
27. National Institute for Standards and Technology. "The NIST Cybersecurity Framework 2.0: Initial Public Draft." NIST CSWP 29, Gaithersburg, MD, https://nvlpubs.nist.gov/nistpubs/CSWP/NIST.CSWP.29.ipd.pdf, 8 August 2023.
28. Sonatype, Inc. "9th Annual State of the Software Supply Chain." *Sonatype*, https://www.sonatype.com/state-of-the-software-supply-chain/introduction, 3 October 2023.
29. Zahan, N., T. Zimmermann, P. Godefroid, B. Murphy, C. Maddila, and L. Williams. "What Are Weak Links in the npm Supply Chain?" *Proceedings of the 44th International Conference on Software Engineering: Software Engineering in Practice*, pp. 331–340, May 2022.
30. Okafor, C., T. R. Schorlemmer, S. Torres-Arias, and J. C. Davis. "Sok: Analysis of Software Supply Chain Security by Establishing Secure Design Properties." *Proceedings of the 2022 ACM Workshop on Software Supply Chain Offensive Research and Ecosystem Defenses*, pp. 15–24, November 2022.
31. Ellison, R. J., J. B. Goodenough, C. B. Weinstock, and C. Woody. "Evaluating and Mitigating Software Supply Chain Security Risks." CMU/SEI-2010-TN-016, Carnegie Mellon University Software Engineering Institute, Pittsburgh, PA, 1 May 2010.
32. Cybersecurity and Infrastructure Security Agency. "Defending Against Software Supply Chain Attacks." Arlington, VA, https://www.cisa.gov/sites/default/files/publications/defending_against_software_supply_chain_attacks_508_1.pdf, April 2021.
33. Simpson, S. (editor). "Fundamental Practices for Secure Software Development: A Guide to the Most Effective Secure Development Practices in Use Today." Software Assurance Forum for Excellence in Code, https://safecode.org/publication/SAFECode_Dev_Practices1008.pdf, 8 October 2008.
34. McGraw, G. "Building Security in Maturity Model." https://owasp.org/www-pdf-archive/Bsimm09.pdf, October 2009.
35. U.S. Department of Homeland Security, Software Assurance (SwA) Processes and Practices Working Group. "Process Reference Model for Assurance Mapping to CMMI-DEV V1.2." 23 June 2008.
36. Open Web Applications Security Project. "Software Assurance Maturity Model." https://wiki.owasp.org/index.php/OWASP_SAMM_Project, 2023.
37. U.S. Executive Office of the President. "Improving Critical Infrastructure Cybersecurity." *National Archives*, Executive Order 13636, 78 FR 11739, https://www.federalregister.gov/documents/2013/02/19/2013-03915/improving-critical-infrastructure-cybersecurity, 12 February 2013.

REFERENCES, *continued*

38. National Institute for Standards and Technology. "NIST Drafts Major Update to Its Widely Used Cybersecurity Framework." *NIST News*, https://www.nist.gov/news-events/news/2023/08/nist-drafts-major-update-its-widely-used-cybersecurity-framework, 8 August 2023.

39. Chandramouli, R., F. Kautz, and S. T. Arias. "Strategies for the Integration of Software Supply Chain Security in DevSecOps CI/CD Pipelines." *NIST/CSRC*, NIST SP 800-204D, initial public draft, https://csrc.nist.gov/pubs/sp/800/204/d/ipd, 30 August 2023.

40. Center for Internet Security. "CIS Critical Security Control 4: Secure Configuration of Enterprise Assets and Software." *CIS*, CIS CSC 4, https://www.cisecurity.org/controls/secure-configuration-of-enterprise-assets-and-software, November 2022.

41. Information Systems Audit and Control Association. "COBIT: An ISACA Framework." Schaumburg, IL, https://www.isaca.org/resources/cobit, accessed December 2023.

42. International Society of Automation. "Security for Industrial Automation and Control Systems Part 2-1: Establishing an Industrial Automation and Control Systems Security Program." ISA-62443-2-1:2009, Triangle Park, NC, https://www.isa.org/products/isa-62443-2-1-2009-security-for-industrial-automat, 2009.

43. International Organization for Standardization and International Electrotechnical Commission. "Information Technology: Security Techniques—Information Security Management Systems—Requirements." ISO/IEC 27001:2013, https://webstore.ansi.org/preview-pages/ISO/preview_ISO+IEC+27001-2013.pdf, 1 October 2013.

44. National Institute for Standards and Technology. "Security and Privacy Controls for Information Systems and Organizations." NIST SP 800-53, Revision 4, *NIST*, https://csrc.nist.gov/pubs/sp/800/53/r4/final, updated 30 April 2013.

45. International Society of Automation. "Security for Industrial Automation and Control Systems Part 3-3: System Security Requirements and Security Levels." ANSI/ISA-62443-3-3:2013, Triangle Park, NC, https://www.isa.org/products/ansi-isa-62443-3-3-99-03-03-2013-security-for-indu, 2013.

46. Center for Internet Security. "Incident Response and Management." *CSF Tools*, CIS CSC 19, https://csf.tools/reference/critical-security-controls/version-7-1/csc-19/, accessed December 2023.

47. Center for Internet Security. "Penetration Tests and Red Team Exercises." *CSF Tools*, CIS CSC 20, https://csf.tools/reference/critical-security-controls/version-7-1/csc-20/, accessed December 2023.

48. National Institute for Standards and Technology. "National Online Informative Reference Program." *NIST*, https://csrc.nist.gov/projects/olir/informative-reference-catalog#/, accessed December 2023.

49. U.S. Executive Office of the President. "Improving the Nation's Cybersecurity." *National Archives*, Executive Order 14028, 86 FR 26633, https://www.federalregister.gov/documents/2021/05/17/2021-10460/improving-the-nations-cybersecurity, 12 May 2021.

50. in-toto Authors. "What Is in-toto?" *in-toto*, https://in-toto.io/in-toto/, accessed 20 July 2023.

51. National Institute for Standards and Technology. "Strategies for the Integration of Software Supply Chain Security in DevSecOps CI/CD Pipelines: NIST SP 800-204D, ipd Available for Comment." *NIST*, https://csrc.nist.gov/News/2023/nist-sp-800-204d-ipd-available-for-comment, 30 August 2023.

52. The Linux Foundation. "What Is SLSQ? Supply-Chain Levels for Software Artifacts, or SLSA (Salsa)." *SLSA*, https://slsa.dev/, 2023.

53. Shetty, S. "Assured Cyber Supply Chain Provenance Using Permissioned Blockchain." *I: The Grainger College of Engineering Information Trust Institute*, University of Illinois Urbana-Champaign, https://iti.illinois.edu/credc/researchactivity/assured-cyber-supply-chain-provenance-using-permissioned-blockchain, 2020.

54. Shetty, S., C. A. Kamhoua, and L. L. Njilla (editors). *Blockchain for Distributed Systems Security*. Hoboken, NJ: John Wiley & Sons, April 2019.

55. Bandara, E., S. Shetty, A. Rahman, and R. Mukkamala. "Let'sTrace—Blockchain, Federated Learning and TUF/In-ToTo Enabled Cyber Supply Chain Provenance Platform." Presented at the 2021 IEEE Military Communications Conference (MILCOM), San Diego, CA, https://ieeexplore.ieee.org/document/9653024, 29 November–2 December 2021.

56. Nath, P., J. R. Mushahary, U. Roy, M. Brahma, and P. K. Singh. "AI and Blockchain-Based Source Code Vulnerability Detection and Prevention System for Multiparty Software Development." *Computers and Electrical Engineering*, vol. 106, https://doi.org/10.1016/j.compeleceng.2023.108607, March 2023.

REFERENCES, *continued*

57. Tang, G., L. Meng, H. Wang, S. Ren, Q. Wang, L. Yang, and W. Cao. "A Comparative Study of Neural Network Techniques for Automatic Software Vulnerability Detection." 2020 International Symposium on Theoretical Aspects of Software Engineering (TASE), pp. 1–8, Hangzhou, China, https://doi.org/10.1109/TASE49443.2020.00010, 2020.

58. Partenza, G., T. Amburgey, L. Deng, J. Dehlinger, and S. Chakraborty. "Automatic Identification of Vulnerable Code: Investigations With an AST-Based Neural Network." 2021 IEEE 45th Annual Computers, Software, and Applications Conference (COMPSAC), pp. 1475–1482, Madrid, Spain, https://doi.org/10.1109/COMPSAC51774.2021.00219, 2021.

59. GitHub, Inc. "CodeQL." *GitHub*, https://codeql.github.com/, accessed 1 December 2023.

60. Facebook, Inc. "Infer: A Tool to Detect Bugs in Java and C/C++?Objective-C Code Before It Ships." *Infer*, https://fbinfer.com/, 1 December 2023.

61. Snyk Limited. "Snyk Powered by DeepCode AI." *Snyk*, https://snyk.io/platform/deepcode-ai/, 1 December 2023.

62. Microsoft. "Type Less, Code More: Visual Studio IntelliCode Brings AI Assistance Directly Into Your Personal Development Flow." *Microsoft/Visual Studio*, https://visualstudio.microsoft.com/services/intellicode/, 1 December 2023.

This Page Intentionally Left Blank

This Page Intentionally Left Blank

This Page Intentionally Left Blank

APPLICATIONS OF ARTIFICIAL INTELLIGENCE (AI) FOR PROTECTING SOFTWARE SUPPLY CHAINS (SSCS) IN THE DEFENSE INDUSTRIAL BASE (DIB)

Abdul Rahman

CSIAC-BCO-2024-499