White Paper

Reducing Cyber Attack Risks Within Software Development Lifecycle Management

ATARC Software Factory Working Group

September 2023

ATARC

Advanced Technology Academic Research Center

Acknowledgements

ATARC would like to take this opportunity to recognize the following Software Factory Working Group members for their contributions:

Stephen King, *Internal Revenue Service (IRS)*
Annette Mitchell, *Internal Revenue Service (IRS)*
Nicholas Weekes, *Internal Revenue Service (IRS)*
Dennis Jerome, *Internal Revenue Service (IRS)*
Claire Bailey, *Veracode*
Eric Florence, *Veracode*
David Wray, *Microfocus*
John Cavanaugh, *Internet Infrastructure Services Corporation*
Robert Ficcaglia, *Cloud Native Computing Foundation (CNCF)*
Al Nieves, *Aqua Security*
Jean-Paul Bergeaux, *Guidepoint*
Hasan Yasar, *Software Engineering Institute | Carnegie Mellon University*
Brian Gallagher, *CodeLock, Inc.*

Table of Contents

Introduction

Increasing cyber-attacks necessitate security improvement and vulnerability reductions to minimize the threats and provide continued government operations. This paper explores cyber risks in the software development process.

We will look at the potential risks and widespread consequences of recent cyber-attacks on the software we use. We'll also discuss strategies and practical steps that can be taken to enhance security when developing and implementing software in government organizations.

Accepted Deployment Patterns

Government entities can leverage existing platforms that will provide secured deployment environments. For example, FedRAMP provides a pre-approved methodology and Authority to Operate (ATO) for software that has met the FedRAMP security requirements.
Additional software development lifecycle application types for Federal Government include:

- General Service Administration GSA FedRAMP AppV (Authorized Vendor)
- GSA FedRAMP modification
- Direct Vendor Purchase
- Direct Vendor Product Modification
- Internal Development Government off the Shelf (GOTs)
- Internal Development with 3rd Party resources

Government may have both on-premesis and cloud software development environments. The best practices for secured software development should require a differentiation of services and components. There are software development services and software development components for which industry must comply through automation.

There are also specific security standards for services provided by industry, including CISA standard for the software supply chain and the software supply chain maturity model. Across federal government, there are multiple compliance requirements that exist today. Each entity, whether public or industry software, must refer to the compliance guidelines and comply as required. The overall goal is to not recreate additional guidelines, but to consolidate for maximum security compliance.

Automation can no longer be viewed as an optional feature; it is now a must. The increased demand for secure code deployment has led both businesses and government organizations to realize that relying simply on manual processes is no longer sufficient to meet the growing need for secure code deployment. Embracing automation has the potential to significantly decrease the time needed to obtain authorization. On the other hand, using manual procedures frequently causes the Authority To Operate (ATO) period to be far longer than is acceptable.

Overview of Recent Software Supply Chain Attacks and Risks

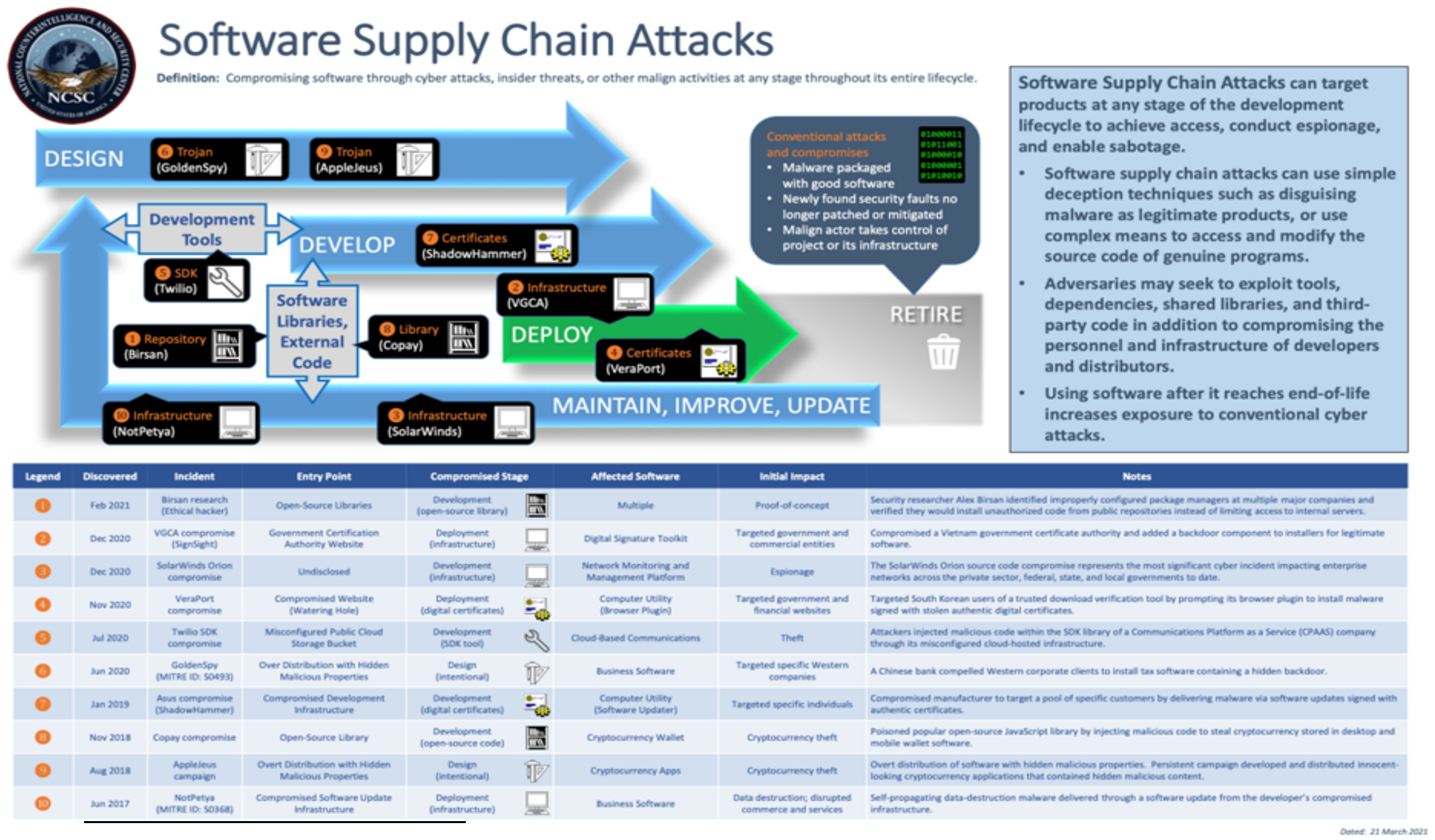
Threat actors are highly skilled at getting beyond cybersecurity safeguards by hacking into networks and using customer-facing applications and APIs to get important data. Potential assaults now encompass more than just concentrating on network-connected gadgets. It now includes all the software-related flaws and vulnerabilities that are present but unresolved in a company's software portfolio. In other words, the danger extends beyond hardware and includes unresolved problems with the software a company utilizes.

Adversaries use attack campaigns for access, espionage, and destruction. They target software supply chains to gain stealthy and persistent access to secured systems and networks. These attacks enable operations ranging from the targeting of specific victims to indiscriminate attacks on connected networks and critical infrastructure. Software supply chain assaults can be used for data alteration or destruction as well as espionage, which involves the covert collection of sensitive information. These attacks create a perilous scenario where access for future attacks is established in a manner that is challenging to detect. In essence, they not only compromise immediate security but also pave the way for subsequent, hard-to-spot intrusions. Improved cybersecurity policies across most networks and computers have made software supply chain attack vectors increasingly attractive because many software development and distribution channels lack sufficient protections. There can be software flaws and vulnerabilities in all software, ranging from First (1st) party software, open-source, containers, and Infrastructure as code (AC).

Government utilizes multiple software environments based on the classification of data and the end user accessing the system. The Department of Defense (DoD) and intelligence community requires information classification applications ranging from public to top secret data, thus requiring stringent application security measures. For cloud services, this translates to FedRAMP Moderate for Impact Level 2 and High and for IL4 and IL5 platforms. A citizen facing system may not require highly secure processing platforms. In addition to platforms, Government may choose to develop software and write programs internally with government employees or system integrators. Government may choose to leverage low-code no-code services with integration to government databases or they may use a custom off-the-shelf solution. Application security is required regardless of the platforming or security requirements. Government services must always continually secure their software development lifecycle.

The question posed to Government agencies is "How does Government address the risk, understand attack vectors, and implement secure application development practices to meet today's cybersecurity threats?" Table 1[1] summarizes the Software Supply chain Attacks and how flaws/vulnerability could be detected at different points in the software delivery process.

Table 1


NCSC
NATIONAL COUNTERINTELLIGENCE AND SECURITY CENTER
DESIGN
6 Trojan (GoldenSpy)
9 Trojan (AppleJeus)
Development Tools
DEVELOP
7 Certificates (ShadowHammer)
5 SDK (Twilio)
Software Libraries, External Code
1 Repository (Birsan)
8 Library (Copay)
2 Infrastructure (VGCA)
DEPLOY
4 Certificates (VeraPort)
10 Infrastructure (NotPetya)
3 Infrastructure (SolarWinds)
MAINTAIN, IMPROVE, UPDATE
RETIRE
Conventional attacks and compromises
Malware packaged with good software
Newly found security faults no longer patched or mitigated
Malign actor takes control of project or its infrastructure

Software Supply Chain Attacks

Definition: Compromising software through cyber attacks, insider threats, or other malign activities at any stage throughout its entire lifecycle.

Software Supply Chain Attacks can target products at any stage of the development lifecycle to achieve access, conduct espionage, and enable sabotage.

- **Software supply chain attacks can use simple deception techniques such as disguising malware as legitimate products, or use complex means to access and modify the source code of genuine programs.**
- **Adversaries may seek to exploit tools, dependencies, shared libraries, and third-party code in addition to compromising the personnel and infrastructure of developers and distributors.**
- **Using software after it reaches end-of-life increases exposure to conventional cyber attacks.**

Legend	Discovered	Incident	Entry Point	Compromised Stage	Affected Software	Initial Impact	Notes
1	Feb 2021	Birsan research (Ethical hacker)	Open-Source Libraries	Development (open-source library)	Multiple	Proof-of-concept	Security researcher Alex Birsan identified improperly configured package managers at multiple major companies and verified they would install unauthorized code from public repositories instead of limiting access to internal servers.
2	Dec 2020	VGCA compromise (SignSight)	Government Certification Authority Website	Deployment (infrastructure)	Digital Signature Toolkit	Targeted government and commercial entities	Compromised a Vietnam government certificate authority and added a backdoor component to installers for legitimate software.
3	Dec 2020	SolarWinds Orion compromise	Undisclosed	Development (infrastructure)	Network Monitoring and Management Platform	Espionage	The SolarWinds Orion source code compromise represents the most significant cyber incident impacting enterprise networks across the private sector, federal, state, and local governments to date.
4	Nov 2020	VeraPort compromise	Compromised Website (Watering Hole)	Deployment (digital certificates)	Computer Utility (Browser Plugin)	Targeted government and financial websites	Targeted South Korean users of a trusted download verification tool by prompting its browser plugin to install malware signed with stolen authentic digital certificates.
5	Jul 2020	Twilio SDK compromise	Misconfigured Public Cloud Storage Bucket	Development (SDK tool)	Cloud-Based Communications	Theft	Attackers injected malicious code within the SDK library of a Communications Platform as a Service (CPAAS) company through its misconfigured cloud-hosted infrastructure.
6	Jun 2020	GoldenSpy (MITRE ID: S0493)	Over Distribution with Hidden Malicious Properties	Design (intentional)	Business Software	Targeted specific Western companies	A Chinese bank compelled Western corporate clients to install tax software containing a hidden backdoor.
7	Jan 2019	Asus compromise (ShadowHammer)	Compromised Development Infrastructure	Development (digital certificates)	Computer Utility (Software Updater)	Targeted specific individuals	Compromised manufacturer to target a pool of specific customers by delivering malware via software updates signed with authentic certificates.
8	Nov 2018	Copay compromise	Open-Source Library	Development (open-source code)	Cryptocurrency Wallet	Cryptocurrency theft	Poisoned popular open-source JavaScript library by injecting malicious code to steal cryptocurrency stored in desktop and mobile wallet software.
9	Aug 2018	AppleJeus campaign	Overt Distribution with Hidden Malicious Properties	Design (intentional)	Cryptocurrency Apps	Cryptocurrency theft	Overt distribution of software with hidden malicious properties. Persistent campaign developed and distributed innocent-looking cryptocurrency applications that contained hidden malicious content.
10	Jun 2017	NotPetya (MITRE ID: S0368)	Compromised Software Update Infrastructure	Deployment (infrastructure)	Business Software	Data destruction; disrupted commerce and services	Self-propagating data-destruction malware delivered through a software update from the developer's compromised infrastructure.

Dated: 21 March 2021

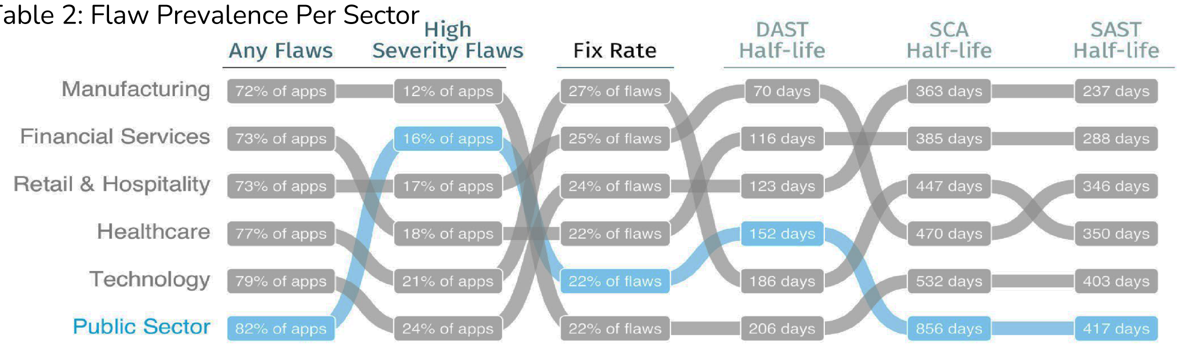
[1] https://www.dni.gov/files/NCSC/documents/supplychain/Software-Supply-Chain-Attacks.pdf

Attack Vectors During the Software Development Process

In Veracode's State of Software Security report, where 759,000+ applications were scanned throughout 2022, the application flaw/vulnerability attack surface is prevalent. Tables 2 and 3[2] showcase that flaws exist and must be remediated regardless of the industry.

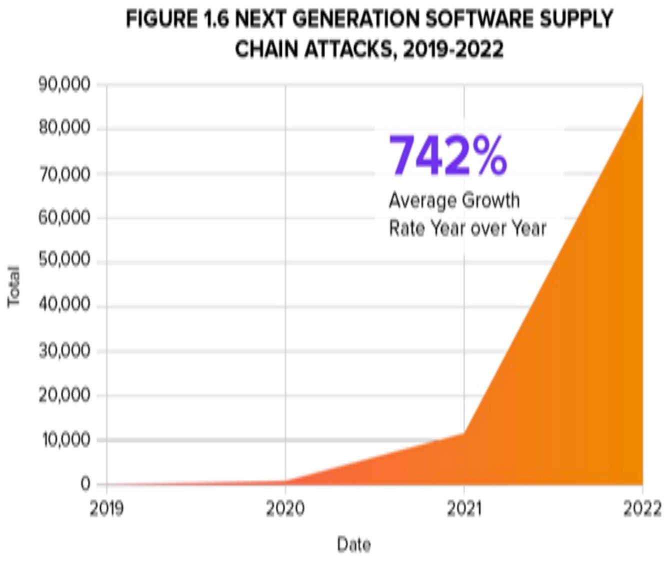
In summary, industry data indicates variables of application security scanning and can identify different types of flaws and vulnerabilities. By embedding a quality gate within the software development process, entities can detect vulnerabilities within the DevSecOps pipelines versus in production environments.

Table 2: Flaw Prevalence Per Sector



	Any Flaws	High Severity Flaws	Fix Rate	DAST Half-life	SCA Half-life	SAST Half-life
Manufacturing	72% of apps	12% of apps	27% of flaws	70 days	363 days	237 days
Financial Services	73% of apps	16% of apps	25% of flaws	116 days	385 days	288 days
Retail & Hospitality	73% of apps	17% of apps	24% of flaws	123 days	447 days	346 days
Healthcare	77% of apps	18% of apps	22% of flaws	152 days	470 days	350 days
Technology	79% of apps	21% of apps	22% of flaws	186 days	532 days	403 days
Public Sector	82% of apps	24% of apps	22% of flaws	206 days	856 days	417 days

Table 3



A recognizable pattern in the occurrence of faults within various sectors is revealed using a thorough statistical analysis covering a large volume of code. It is interesting to observe that the public sector takes the top spot, with a significant 81.9% incidence of faults, of which a noteworthy 77.4% fall under the OWASP Top 10 vulnerabilities. This percentage of defects emerges as a dramatic illustration of the hazards associated with modern federal services when combined with the recognized attack vectors that target the Federal Government. This information highlights the urgent need for increased security and monitoring inside government activities.

[2] *https://info.veracode.com/report-state-of-software-security-2023.html*

Recent Software Attacks

High profile industry software attacks are indicators of the criticality of securing the software supply chain. Examples of recent attacks include:

- Chinese Intelligence Attack, July 2023[3]: Chinese intelligence hacked into Microsoft email accounts belonging to two dozen government agencies, including the State Department.
- W4SP Copycats Continue to Infiltrate PyPi (Python Package Index) Registry, March 2023[4]: A series of malicious packages uploaded to the PyPi registry have been identified as information stealers resembling the popular W4SP stealer. These copycats pose a serious threat to developers who may inadvertently install them, compromising the open-source software supply chain.
- Microsoft-Helper Package Reveals Copycat Info-Stealer March 2023[4]: The Microsoft-helper package on PyPi is a malicious package designed to deploy malware when developers run pip install. It downloads a remote script containing a second-stage payload, which exfiltrates sensitive information through a Discord webhook.
- Open AI Data Breach Traced to Unpatched Redis Vulnerability, March 2023[4]: An unpatched software bug/vulnerability in an open-source component called Redis led to a data breach at Open AI. The incident exposed subscribers' payment-related info and user' chat queries and resulted in Italy becoming the first Western country to ban ChatGPT.
- December 2022[5]: Chinese government-linked hackers stole at least $20 million in COVID-19 relief funds from the U.S. government, including Small Business Administration loans and unemployment insurance money. The U.S. Secret Service announced they retrieved half of the stolen funds thus far.
- November 2022[5]: Suspected Chinese-linked hackers carried out an espionage campaign on public and private organizations in the Philippines, Europe and the United States since 2021. The attacks used infected USB drives to deliver malware to the organizations.

[3] https://www.cnbc.com/2023/07/12/us-government-emails-compromised-by-china-based-espionage-group.html

[4] https://www.sonatype.com/state-of-the-software-supply-chain/introduction

[5] https://www.csis.org/programs/strategic-technologies-program/archives/survey-chinese-espionage-united-states-2000#:~:text=June%202022%3A%20The%20FBI%2C%20National,providers%20since%20at%20least%202020

- June 2022[5]: The FBI, National Security Agency (NSA) and CISA announced that Chinese state-sponsored hackers targeted and breached major telecommunications companies and network service providers since at least 2020.
- March 2022[5]: Hackers linked to the Chinese government penetrated the networks belonging to government agencies of a least six different U.S. states in an espionage operation. Hackers took advantage of the Log4j vulnerabilities to access the networks, in addition to several other vulnerable internet-facing web applications.

Unique Government Software Development Challenges

Government application development relies upon decades of custom software development. Spanning the decades are a variety of languages, platforms, tools, and operating systems, thus creating an infinite loop of knowledge to maintain the software. The impact of technical debt and lack of standardization also impacts software supply chain risk. The ability to produce software can occasionally become segmented, resulting in silos, even within a single agency. Different program or project teams, as well as external support contractors, may exhibit these silos. The many entities involved in software development may find it difficult to communicate, collaborate, and coordinate effectively as a result of this fragmentation.

Government application development environments may have few enterprise services for software developers, DevOps and Security/Operations teams. This creates poor visibility, governance and monitoring of software security metrics. Lack of centrally located Open-Source software libraries and tools for CICD Automation impacts the efficiency and security of the application development. Without standardization and security assessment of API's leveraged for inter-agency information sharing, agencies may not fully comprehend the inherent risk of an unsigned API for a lack of a Software Bill of Materials to truly understand risk. There should be an automated security guardrail at every stage of the CICD pipeline.

The Federal Government led by the Executive Office of the President and the Federal CISO have identified key components to secure the software supply chain. It has been recognized globally that without securing this critical supply chain, the global economy is at risk. Multiple executive orders and memorandums address this as our nation strives to adopt secure software development. Inherent in that journey to secure software is the partnership and compliance by industry solutions. CISA (Cybersecurity and Infrastructure Security Agency**)** has drafted an attestation form that has currently been distributed for comment. This attestation will provide both government and private industry the opportunity to accurately reflect their adherence to government security standards.

- CISA provides the policy for federal government entities as directed by the executive office of the president. CISA also enforces compliance with federal cybersecurity standards.
- GSA provides the procurement path for government entities to purchase software that has an attestation of meeting government security standards.
- NIST provides the detailed technical requirements and standards for government software security and secure software development lifecycles.
- Congress enacts laws to require secure software development lifecycles.

Overarching the United States cybersecurity strategy are the executive orders and guidance from the executive office of the president. The following governing documents are:

- Executive Order 14028
- CISA Cybersecurity Strategic Plan FY 2024-2026[6]
- President's Implementation Plan M-22-18
- White House Strategic Software Security Plan[7]
- NIST Source Code Security Analyzers[8]
- White House Cybersecurity Strategy[9]

[6] https://www.cisa.gov/sites/default/files/2023-08/FY2024-2026_Cybersecurity_Strategic_Plan.pdf

[7] https://www.whitehouse.gov/wp-content/uploads/2023/06/M-23-16-Update-to-M-22-18-Enhancing-Software-Security.pdf

[8] https://www.nist.gov/itl/ssd/software-quality-group/source-code-security-analyzers

[9] https://www.whitehouse.gov/wp-content/uploads/2023/07/National-Cybersecurity-Strategy-Implementation-Plan-WH.gov_.pdf

Industry and Government Recommendations and Frameworks

Critical software systems and the supply chains that deliver them should be designed, and evaluated, with security considerations relevant to confidentiality, integrity, and availability. If a critical system has not been initially designed with security at the forefront, then security assessment and mitigation activities should begin as early as possible to minimize risk over time. This can be accomplished through the implementation and standardization of a software development lifecycle. Government also has the opportunity to leverage GSA secure products through GSA FedRAMP compliance. This strategy move toward the early integration of security measures is based on the idea of foresight, which recognizes that resolving weaknesses and bolstering defenses at the outset helps fend off potentially rising threats and vulnerabilities as the system evolves.

Government can choose to adopt software development practices and frameworks by leveraging information sources, standards, and best practices. These are available through the National Institute of Standards and Technology (NIST), the Common Criteria ISO/IEC 15408, the Collection of guides from Defense Information Systems Agency (DISA), Security Technical Implementation Guide (STIG), OASIS SARIF, PCI SSF, and OWASP ASVS.

When an agency opts to embark on the development or maintenance of their own software, they not only take on the responsibility but also recognize the associated risks. This calls for ensuring that the software is developed and kept up to date in a secure manner. In contrast, using federally endorsed frameworks like FedRAMP offers a method to risk reduction should an agency decide to purchase software created by outside parties. These frameworks offer a structured setting that might aid in reducing potential risks. It emphasizes the need of their proactive oversight and watchful guardianship that the agency remains the guardian of its risk posture regardless of whether the software is internally developed or obtained from an outside source.

Executive Order 14028[10], titled "Improving the Nation's Cybersecurity," signed by President Joe Biden on May 12, 2021, aims to enhance the cybersecurity posture of the

[10] https://www.govinfo.gov/content/pkg/FR-2021-05-17/pdf/2021-10460.pdf

United States. While the executive order encompasses various aspects of cybersecurity, including threat vectors in application lifecycles, it primarily focuses on improving the nation's overall cybersecurity resilience. Specifically, the impact of Executive Order 14028 on threat vectors in application lifecycles. The primary focus on secure software development practices is the adoption of secure software development practices throughout the application lifecycle. It directs federal agencies to develop and enforce baseline security standards for software purchased or developed by the government. These standards address both the design and implementation of software, ensuring that security considerations are integrated from the initial stages of development.

Executive Order 14028 aims to strengthen the cybersecurity posture of the United States, and its impact on threat vectors in application lifecycles is significant. By emphasizing secure software development practices, supply chain security, vulnerability disclosure programs, Zero Trust Architecture, and incident response capabilities, the Executive Order helps reduce the potential threats and vulnerabilities that can arise during the development, deployment, and maintenance of applications.

There are several software lifecycle management and software development frameworks that are widely adopted across government. Regardless of methodology (agile, waterfall, etc.), the most important aspect is to have a secure software delivery lifecycle.

Key components of a secure software delivery lifecycle are:

1. **Requirements and Design:** Security considerations are identified and incorporated into the initial software requirements and design phase. Threat modeling techniques may be employed to identify potential security risks and define appropriate security controls.

2. **Secure Coding:** Developers follow secure coding practices, such as input validation, output encoding, and secure handling of sensitive data. Secure coding guidelines and best practices are adhered to in order to minimize vulnerabilities.

3. **Testing and Verification:** Security testing is conducted throughout the development lifecycle. This includes activities such as static code analysis, dynamic application security testing (DAST), penetration testing, and vulnerability scanning.

These tests help identify security weaknesses and vulnerabilities that need to be addressed.

4. **Security Review and Assessment:** A formal security review is conducted to assess the security posture of the software. This may involve external security experts or internal security teams reviewing the architecture, code, and configuration for potential vulnerabilities.

5. **Deployment and Operations:** Security controls, such as access controls, encryption, and logging, are implemented during deployment and operation phases. Regular monitoring and incident response capabilities are established to detect and respond to security incidents.

6. **Maintenance and Updates:** The software is regularly updated and patched to address newly discovered security vulnerabilities. Security updates are deployed promptly to ensure the ongoing security of the software.

The Software Security Development Lifecycle (SSDLC) aims to integrate security considerations throughout the entire software development process rather than treating security as an afterthought. By incorporating security from the early stages, organizations can reduce the likelihood of security breaches and deliver software that meets high-security standards.

SLSA

Supply-chain Levels for Software Artifacts, or SLSA[11] ("salsa") is a security framework, a checklist of standards and controls to prevent tampering, improve integrity, and secure packages and infrastructure. SLSA is a set of incrementally adoptable guidelines for supply chain security, established by industry consensus. Producers can follow SLSA's guidelines to make their software supply chain more secure, and consumers can use SLSA to make decisions about whether to trust a software package.

[11] https://slsa.dev/spec/v1.0/

SLSA provides:

- A common vocabulary to talk about software supply chain security
- A way to secure your incoming supply chain by evaluating the trustworthiness of the artifacts you consume
- An actionable checklist to improve your own software's security
- A way to measure your efforts toward compliance

SLSA provides protection against tampering along the supply chain to consumers, both reducing insider risk and increasing confidence that the software produced reaches consumers as intended.

Software consumers, such as a development team using open source packages, a government agency using vendored software, or a CISO judging organizational risk use SLSA as a way to judge the security practices of the software they rely on and the integrity of that software. Adoption of SLSA enables a secure software supply chain between Infrastructure providers, who provide infrastructure such as an ecosystem package manager, build platform, or CI/CD platform and software consumers.

A SLSA track focuses on a particular aspect of a supply chain, such as the Build Track. SLSA v1.0 consists of only a single track (build), but future versions of SLSA will add tracks that cover other parts of the software supply chain, such as how source code is managed.
Within each track, ascending levels indicate increasingly hardened security practices. Higher levels provide better guarantees against supply chain threats but come at higher implementation costs. Lower SLSA levels are designed to be easier to adopt, but with only modest security guarantees. SLSA 0 is sometimes used to refer to software that doesn't yet meet any SLSA level. Currently, the SLSA Build Track encompasses Levels 1 through 3, but higher levels will be possible in future revisions.

The combination of tracks and levels offers an easy way to discuss whether software meets a specific set of requirements. By referring to an artifact as meeting SLSA Build Level 3, for example, you're indicating in one phrase that the software artifact was built following a set of security practices that industry leaders agree protect against supply chain compromises.

SLSA's framework addresses every step of the software supply chain - the sequence of steps resulting in the creation of an artifact. SLSA represents a supply chain as a collection of sources, builds, dependencies, and packages. One artifact's supply chain is a combination of its dependencies' supply chains plus its own sources and builds.

There are several areas outside SLSA's current framework that are important to consider together with SLSA such as:

- Code quality: SLSA does not tell if secure coding practices were followed when writing the code.
- Producer trust: SLSA does not address organizations that intentionally produce malicious software, but it can reduce insider risks within a trusted organization. SLSA's Build Tract protects against tampering during or after the build, and future SLSA tracks intend to protect against unauthorized modifications of source code and dependencies.
- Transitive trust for dependencies: The SLSA level of an artifact is independent of the level of its dependencies. SLSA can be used recursively to judge an artifact's dependencies on their own, but there is currently no single SLSA level that applies to both an artifact and its transitive dependencies together.

Heroku Framework

The uniform implementation of a standard application development framework is essential to the notion of secure software development, whether it is applied to on-premises infrastructures or cloud platforms. This strategy makes sure that the source code possesses security features that go beyond the underlying platform, allowing it to function securely across a wide range of contexts. A unified security posture is embodied in the seamless portability of software across platforms.

The Heroku framework, a cloud application platform, has both positive and negative impacts on application security. Here are some ways in which the Heroku framework can impact application security:

1. **Infrastructure Security**: Heroku provides a secure infrastructure for hosting applications. It manages the underlying infrastructure, including servers, networks, and data centers, which can help mitigate certain security risks associated with managing infrastructure on-premises. Heroku's infrastructure is designed to be highly available, scalable, and resilient, which can contribute to the overall security of the hosted applications.

2. **Platform Security:** Heroku takes care of platform-level security measures, such as securing the operating system, patching vulnerabilities, and managing network security. This helps protect applications from common security threats that target the underlying platform. Heroku also ensures compliance with industry standards and

regulations, which can be beneficial for applications that have specific security requirements.

3. **Access Control and Authentication:** Heroku provides access control mechanisms to manage user access to applications and resources. It supports various authentication methods, including multi-factor authentication (MFA), OAuth, and identity providers like Okta and Active Directory. These features help enforce proper access controls and reduce the risk of unauthorized access to applications and sensitive data.

4. **Application Isolation:** Heroku uses a container-based architecture to isolate applications from one another. Each application runs in its own isolated environment, which helps prevent cross-application attacks and limits the impact of security breaches. This isolation provides an additional layer of security for applications hosted on the Heroku platform.

5. **Secure Deployment:** Heroku provides secure deployment mechanisms that ensure the integrity and authenticity of application updates. It supports version control systems like Git and provides secure deployment options, such as encrypted connections (HTTPS) and secure shell (SSH) access. These features help protect applications during the deployment process and reduce the risk of unauthorized code changes or tampering.

Despite these advantages, it's important to stress that the field of application security is a joint effort that goes beyond the limitations of any single institution. Within this context, a mutually beneficial cooperation develops between the Heroku platform and the application developers, with each party sharing responsibility for bolstering the software's defenses. Developers need to implement secure coding practices, conduct regular security testing, and address application-specific vulnerabilities. Additionally, while Heroku provides a secure environment, the security of the application itself, including the code, data handling, and user authentication, remains the responsibility of the application owner.

In summary, the Heroku framework can positively impact application security by providing a secure infrastructure, managing platform-level security, enforcing access controls, isolating applications, and offering secure deployment options. However, developers and application owners must also take proactive measures to ensure the security of their applications within the Heroku environment.

Authority to Operate

The National Institute of Standards and Technology[12] (NIST) Risk Management Framework (RMF) enables the Authority to Operate (ATO) process, which is a framework used to assess and authorize the operation of information systems and applications within federal agencies. The ATO process has several impacts on application security and software development:

1. **Security Compliance:** The ATO process requires applications to comply with a set of security controls and guidelines defined by NIST, such as the NIST Special Publication 800-53. These controls cover various aspects of application security, including access control, encryption, vulnerability management, incident response, and more. As a result, the ATO process drives the adoption of security best practices and ensures that applications meet the required security standards.

2. **Risk Assessment:** The ATO process involves a comprehensive risk assessment of the application. This assessment identifies potential vulnerabilities, threats, and risks associated with the application's design, development, and operational aspects. Through this process, security weaknesses and gaps can be identified and addressed, leading to improved application security.

3. **Security Testing and Evaluation:** As part of the ATO process, applications undergo rigorous security testing and evaluation. This includes vulnerability scanning, penetration testing, code review, and other security assessments. These tests help identify vulnerabilities, weaknesses, and potential attack vectors within the application. By uncovering these issues, developers can rectify them and enhance the overall security posture of the application.

4. **Secure Software Development Lifecycle (SDLC):** The ATO process promotes the adoption of a secure software development lifecycle (SDLC) methodology. It encourages developers to integrate security considerations throughout the entire software development process, from requirements and design to deployment and maintenance. This includes secure coding practices, threat modeling, code reviews, and

[12] https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-53r5.pdf

security testing at various stages. As a result, the ATO process helps foster a culture of security-conscious software development.

5. **Continuous Monitoring and Compliance:** Once an application receives an ATO, it is subject to continuous monitoring and compliance requirements. This entails regular security assessments, vulnerability scanning, log analysis, and incident response activities. By continuously monitoring the application's security posture, vulnerabilities and security incidents can be promptly detected and addressed, thereby ensuring ongoing application security.

Overall, the Agency Authority to Operate (ATO) process has a significant impact on application security and software development. It drives compliance with security standards, promotes risk assessment and mitigation, encourages the adoption of secure SDLC practices, and enforces continuous monitoring and compliance. By following the ATO process, organizations can enhance the security of their applications and mitigate potential risks and vulnerabilities throughout the software development lifecycle.

Secure Software Development Innovations

There are several secure software development frameworks available that provide guidelines, best practices, and methodologies to develop secure applications. Here are three popular frameworks and how they differentiate from one another:

1. **Microsoft Secure Development Lifecycle (SDL):** The Microsoft SDL is a framework developed by Microsoft to integrate security practices into the software development process. It consists of a set of security-focused activities and practices that cover the entire software development lifecycle. The SDL emphasizes threat modeling, code analysis, secure coding practices, security testing, and security training for developers. It provides specific guidance and tools tailored for Microsoft technologies and platforms. The SDL is known for its comprehensive approach to application security and its integration with Microsoft development tools.

2. **Open Web Application Security Project (OWASP) Software Assurance Maturity Model (SAMM):** The OWASP SAMM is an open framework that helps organizations assess, formulate, and implement a strategy for software security. It provides a maturity model that guides organizations through different levels of maturity in building secure software. The SAMM framework covers various domains, including governance, design, implementation, verification, and operations. It emphasizes the

importance of security culture, risk assessment, secure architecture, security testing, and secure deployment practices. The SAMM framework is technology-agnostic and can be applied to different software development environments.

3. **Building Security in Maturity Model (BSIMM):** The BSIMM is a framework that focuses on software security initiatives within organizations. It is a descriptive model that captures the activities, practices, and measurements of real-world software security initiatives. The BSIMM framework is based on data collected from various organizations and provides a benchmark for software security practices. It consists of a set of 12 security practices grouped into four domains: governance, intelligence, secure software development lifecycle, and deployment. The BSIMM framework helps organizations understand and improve their software security maturity based on industry-proven practices.

These frameworks differentiate from one another in several ways:

- **Scope:** Each framework may have a different focus and scope. For example, the Microsoft SDL primarily targets Microsoft technologies, while OWASP SAMM and BSIMM are more technology-agnostic and can be applied to various software development environments.
- **Approach:** The frameworks may have different approaches to software security. For instance, the SDL emphasizes secure coding practices and integration with Microsoft development tools, while OWASP SAMM focuses on overall software security strategy and maturity levels. BSIMM captures real-world practices and provides a benchmark for software security initiatives.
- **Guidance and Tools:** The frameworks may provide different guidance, best practices, and tools. For example, the SDL offers specific guidance and tools for Microsoft technologies, while OWASP SAMM provides a broader set of guidelines applicable to different technologies and platforms.
- **Maturity Model:** The frameworks may differ in their maturity models and how they assess and measure software security maturity. OWASP SAMM and BSIMM both provide maturity models, but they may have different domains, activities, and measurements.
- **Application Security Posture/Score:** Industry through automation can now assess the overall security posture of the application. Whether the application is in development or in production, industry tools can spotlight the security vulnerabilities and assess the impact to agency operations. Industry can also provide automation in the remediation process based on machine learning and artificial intelligence.

It is important to note that these frameworks are not mutually exclusive, and organizations can choose to adopt multiple frameworks based on their specific needs and requirements. The selection of a framework depends on factors such as the organization's technology stack, development processes, security goals, and resources available.

Strategies to Improve Resilience in the Software Development Process

Critical to the success of a secure software development lifecycle is the ability to understand the root causes of mission and security failure and identify areas for improvement within the SDLC, CI/CD Automation and tools. Key areas of focus to secure Design, Development, Test/Validate, Deploy, Maintain/Update functions are:

- Leveraging cloud-based repositories for source code, binaries and DevSecOps build environments
- Documenting potential attack vectors and mitigating and/or eliminating them
- Promoting enterprise-wide services for DevSecOps for all high-level attack vectors
- Leveraging zero trust and insider threat modeling for developers and IT administrators
- Improving governance of code, software assurance, open-source governance
- Continuous code signing, credentialing and validation
- New cloud-based service offerings to reduce supply chain threat vectors
- Enterprise Software Assurance, Testing, Source Code Repositories, etc.
- Enterprise Scale Agile Frameworks (SaFE)
- Threat advisory services
- Software signature validation
- Enhanced product capability to support enterprise software development capabilities
- Support for industry standards for software development standards
- Automation for software build process (must be actionable)
- How software developers use request to initiate new code changes to new code repositories
- How industry automates the PR analysis to ensure no new software vulnerabilities are introduced; deployment frequency, code goes through security checks
- 12 factor application[13]

[13] https://12factor.net/

Threat Vector Examples

#	Threat Vector	Security Control
1	Rogue Developer / Employee	Include BG check / others as part of recruitment process (also required by ISO27001) Multiple or paired code reviewers required for deployment
2	Suspicious developers' behavior	Utilize insider threat modeling Analyze user behavior from endpoint audit logs
3	Developers' laptop might be compromised	Laptop must be regularly patched according to MF corporate policy Full device management (no Local Admin permissions, no removable USB devices, runtime malware scanning)
4	Uncontrolled SW installed on the laptop by the developer	Limit the developers' ability to install uncontrolled SW
5	Uncontrolled/Approved IDEs may lead to laptop compromise	Define list of approved IDEs Define policies for an approved IDE
6	Insecure configuration of IDE may lead to compromise	Verify the security of the IDE (i.e., patches, secure configuration, least privileges, logging, auto update) Deploy IDE security plug-ins
7	Uncontrolled code & libs presented by the developer	Scan source code and libs on the developers' laptop for vulnerabilities Scan libs on laptop for non-used libs Package developers – explicitly declare the dependencies and define a lock file Avoid fake commits by signing the commit Build always from source and not from local directory
8	Interface to source control repo might be vulnerable	Client patch updates and auto update Secure configuration of client, verify proper encryption, authentication, authorization, etc.
9	Inability to conduct forensics	Each item above must be configured for proper logging Centralized audit log infrastructure and applications for correlation and searching of logs from endpoints, integration with SEIM

Matrix of Potential Threats, Security Controls, and Steps

The spectrum of threats extends its reach through a myriad of sources and can impact different components of the software development cycle. We have accumulated a matrix of the threat vectors and identified security controls which can be implemented to minimize risk to the agency. The table below identifies two threat vectors, developer and source code control, which should be immediately addressed.

Source Code Control

#	Threat Vector	Security Control
1	Infrastructure related attacks	OS Hardening; patches; NW segmentation; MW scanning; AV; host-based IDS, etc. IAM via SSO Least privileges – who can access, what can they do on the OS level
2	Uncontrolled/Approved source code repos may lead to compromise	Define list of approved source code repos Define requirements & Policy for an approved source code repo Ensure all source code repos have immutable logs and transactions and are being monitored for malicious activity
3	Insecure configuration of source code repo may lead to compromise	Verify the security of the source code repo (i.e., patches, secure configuration, least privileges, logging, auto update) Define proper authorization, authentication, least privileges concept (Zero Trust) Access via VPN only / use 2FA and continuous identity evaluation Enable access only via SSO/SAML Disable usage of 'shared' accounts
4	Insecure code may lead to compromise	Force code review (no self) as condition for push (gap: training on what to look for – i.e., source of code; backdoors, source from non-approved, manifest file changes?) Deploy scanners from feature branch as well as master branch: SAST; encoded; obfuscated; externally originated; MW; behavioral (Old code, dormant developer, etc...); Manifest files must have explicit permissions and auditing for every change
5	Insecure interfaces to/from source control repo might be vulnerable	Define secure interface from "source code control" to push/pull to "automation server/build machine" i.e., specific token, encryption, authentication, authorization, logging, auto update, etc. Enable token renewal
6	Inability to conduct forensics	Each item above must be configured for proper logging to a SEIM
7	Lack of government might lead to system compromise	Conduct PT/security assessment on the source control at defined cadence – TBD (cadence and funding)

Summary

Our recommendation is to require enterprise approaches to DevSecOps services. To do this, you must have a secure platform for your enterprise artifacts, your enterprise source code controls, and leverage enterprise orchestrated and automated services. These automated services must address the application build process, enterprise threat mitigation assessment, and security testing.

Functional and performance testing must be performed on every build deployment and must meet your agency's risk profile. Every build must perform static testing, dynamic testing, and code coverage testing. Testing and provisioning should leverage deployment scripts Infrastructure as Code (IACs). Depending upon the infrastructure for the platform, secure enterprise cloud templates and secure cloud deployment should address your agency environment. For example, Kubernetes space deployments and the security required.

Appendix

For further information, consider the following references contained within NIST's documentation on:

- SSDF and C-SRCM.
- NIST: Mitigating the Risk of Software Vulnerabilities by Adopting a Secure Software
- Development Framework (SSDF)
- BSIMM: Building Security in Maturity Model (BSIMM) Version 11
- BSA: The BSA Framework for Secure Software: A New Approach to Securing the Software
- Lifecycle, Version 1.1
- Institute for Defense Analyses (IDA): State-of-the-Art Resources (SOAR) for Software
- Vulnerability Detection, Test, and Evaluation
- International Organization for Standardization/International Electrotechnical Commission
- (ISO/IEC): Information technology – Security techniques – Application security – Part 1: Overview and concepts, ISO/IEC 27034-1:2011
- Microsoft: Microsoft Security Development Lifecycle
- NIST: Framework for Improving Critical Infrastructure Cybersecurity, Version 1.1

- NIST: SP 800-53 Rev. 5, Security and Privacy Controls for Information Systems and Organizations
- NIST: SP 800-160 Vol. 1, Systems Security Engineering: Considerations for a Multidisciplinary
- Open Web Application Security Project (OWASP): OWASP Application Security Verification Standard 4.0.2
- OWASP: Software Assurance Maturity Model Version 1.5
- Payment Card Industry (PCI) Security Standards Council: Secure Software Lifecycle (Secure
  SLC)
- Requirements and Assessment Procedures Version 1.1
- Software Assurance Forum for Excellence in Code (SAFECode): Fundamental Practices for Secure Software Development: Essential Elements of a Secure Development Lifecycle
- Program, Third Edition
- SAFECode: Managing Security Risks Inherent in the Use of Third-Party Components
- SAFECode: Practical Security Stories and Security Tasks for Agile Development Environments
- SAFECode: Software Integrity Controls: An Assurance-Based Approach to Minimizing Risks in the Software Supply Chain
- SAFECode: Tactical Threat Modeling

Additional Resources:

- Reasonable Accommodation Act Section 508 ***https://www.cisa.gov/sites/default/files/publications/defending_against_software_supply_chain_attacks_508_1.pdf***
- *Secure software development framework (SSDF)*
- *NIST Cyber Supply Chain Risk Management Framework (C-SCRM) and the Secure Software Development Framework (SSDF)*
- *Build Security In Maturity Model (BSIMM) & CMMI, Open Web Application Security Project (OWASP)*
- FITARA Scoring
- Veracode 2023 Annual Report on the State of Application Security - https://info.veracode.com/report-state-of-software-security-2023.html
- Veracode State of Software Report by Sectors - https://www.veracode.com/sites/default/files/pdf/resources/reports/veracode-state-of-software-security-2023-public-sector.pdf