SECURING THE SOFTWARE SUPPLY CHAIN

RECOMMENDED PRACTICES GUIDE FOR

CUSTOMERS

Enduring Security Framework

October 2022

Executive Summary

Cyberattacks are conducted via cyberspace and targets an enterprise's use of cyberspace for the purpose of disrupting, disabling, destroying, or maliciously controlling a computing environment or infrastructure; or destroying the integrity of the data or stealing controlled information.[1]

Recent cyberattacks such as those executed against SolarWinds and its customers, and exploits that take advantage of vulnerabilities such as the Log4j, highlight weaknesses within software supply chains; an issue which spans both commercial and open source software and impacts both private and Government enterprises. Accordingly, there is an increased need for software supply chain security awareness and cognizance regarding the potential for software supply chains to be weaponized by nation state adversaries using similar tactics, techniques, and procedures (TTPs).

In response, the White House released an Executive Order on Improving the Nation's Cybersecurity (EO 14028). EO 14028 establishes new requirements to secure the federal government's software supply chain. These requirements involve systematic reviews, process improvements, and security standards for both software suppliers and developers, in addition to customers who acquire software for the Federal Government.

Similarly, the Enduring Security Framework[2] (ESF) Software Supply Chain Working Panel has established this guidance to serve as a compendium of suggested practices for developers, suppliers, and customer stakeholders to help ensure a more secure software supply chain. This guidance is organized into a three part series: Part 1 of the series focuses on software developers; Part 2 focuses on software suppliers; and Part 3 focuses on software customers.

Customers (acquiring organizations) may use this guidance as a basis of describing, assessing, and measuring security practices relative to the software lifecycle. Additionally, suggested practices listed herein may be applied across the acquisition, deployment, and operational phases of a software supply chain.

The software supplier (vendor) is responsible for liaising between the customer and software developer. Accordingly, vendor responsibilities include ensuring the integrity and security of software via contractual agreements, software releases and updates, notifications, and mitigations of vulnerabilities. This guidance contains recommended best practices and standards to aid suppliers in these tasks.

This document will provide guidance in line with industry best practices and principles which software developers are strongly encouraged to reference. These principles include security requirements planning, designing software architecture from a security perspective, adding security features and maintaining the security of software and the underlying infrastructure (e.g., environments, source code review, testing).

---

[1] Committee on National Security Systems (CNSS)

[2] The ESF is a cross-sector working group that operates under the auspices of Critical Infrastructure Partnership Advisory Council (CIPAC) to address threats and risks to the security and stability of U.S. national security systems. It is comprised of experts from the U.S. government as well as representatives from the Information Technology, Communications, and the Defense Industrial Base sectors. The ESF is charged with bringing together representatives from private and public sectors to work on intelligence-driven, shared cybersecurity challenges.

DISCLAIMER

DISCLAIMER OF ENDORSEMENT

This document was written for general informational purposes only. It is intended to apply to a variety of factual circumstances and industry stakeholder, and the information provided herein is advisory in nature. The guidance in this document is provided "as is." Once published, the information within may not constitute the most up-to-date guidance or technical information. Accordingly, the document does not, and is not intended to, constitute compliance or legal advice. Readers should confer with their respective advisors and subject matter experts to obtain advice based on their individual circumstances. In no event shall the United States Government be liable for any damages arising in any way out of the use of or reliance on this guidance.

Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not constitute or imply its endorsement, recommendation, or favoring by the United States Government, and this guidance shall not be used for advertising or product endorsement purposes. All trademarks are the property of their respective owners.

PURPOSE

NSA, ODNI, and CISA developed this document in furtherance of their respective cybersecurity missions, including their responsibilities to develop and issue cybersecurity recommendations and mitigations. This information may be shared broadly to reach all appropriate stakeholders.

CONTACT

Client Requirements / Inquiries: Enduring Security Framework nsaesf@cyber.nsa.gov

Media Inquiries / Press Desk:

- NSA Media Relations, 443-634-0721, MediaRelations@nsa.gov
- CISA Media Relations, 703-235-2010, CISAMedia@cisa.dhs.gov
- ODNI Media Relations, dni-media@dni.gov

Table of Contents

1 Introduction

Unmitigated vulnerabilities in the software supply chain pose a significant risk to organizations. This series presents actionable recommendations for a software supply chain's development, production and distribution, and management processes to increase the resiliency of these processes against compromise.

All organizations have a responsibility to establish software supply chain security practices to mitigate risks, but the organization's role in the software supply chain lifecycle determines the shape and scope of this responsibility.

Because the considerations for securing the software supply chain vary based on the role an organization plays in the software supply chain, this series presents recommendations geared toward these important roles, namely, developers, suppliers, and customers (or the organization acquiring a software product),

This guidance is organized into a three part series and will be released coinciding with the software supply chain lifecycle. This is Part 3 of the series which focuses on the software customer. Part 1 of the series focused on software developers and Part 2 focused on software suppliers. This series will help foster communication between these three different roles and among cybersecurity professionals that may facilitate increased resiliency and security in the software supply chain process.

In this series, terms such as risk, threat, exploit, and vulnerability are based on descriptions defined in the Committee on National Security Systems Glossary (CNSSI 4009).[3]

1.1 Background

Historically, software supply chain compromises largely targeted commonly known vulnerabilities organizations that were left unpatched. While threat actors still use this tactic to compromise unpatched systems, a new, less conspicuous method of compromise also threatens software supply chains and undermines trust in the patching systems themselves that are critical to guarding against legacy attacks. Rather than waiting for public vulnerability disclosures, threat actors proactively inject malicious code into products that are then legitimately distributed downstream through the global software supply chain. Over the last few years, these next-gen software supply chain compromises have significantly increased for both open source and commercial software products.

Technology consumers generally manage software downloads and broader, more traditional software supply chain activities separately. Considering both the upstream and downstream phases of software as a component of supply chain risk management may help to identify problems and provide a better way forward in terms of integrating activities to achieve systemic security. However, there are also some differences to account for in the case of software products. A traditional software supply chain cycle is from point of origin to point of consumption and generally enables a customer to return a malfunctioning product and confine any impact. In contrast, if a

[3] CNSSI-4009.pdf

software package is injected with malicious code which proliferates to multiple consumers; the scale may be more difficult to confine and may cause an exponentially greater impact.

Common methods of compromise used against software supply chains include exploitation of software design flaws, incorporation of vulnerable third-party components into a software product, infiltration of the supplier's network with malicious code prior to the final software product being delivered, and injection of malicious software that is then deployed by the customer.

Stakeholders must seek to mitigate security concerns specific to their area of responsibility. However, other concerns may require a mitigation approach that dictates a dependency on another stakeholder or a shared responsibility by multiple stakeholders. Dependencies that are inadequately communicated or addressed may lead to vulnerabilities and the potential for compromise.

Areas where these types of vulnerabilities may exist include:

- Undocumented features or high risk functionality,
- Unknown and/or revisions to contractual, functionality, or security assumptions between evaluation and deployment,
- Supplier's change of ownership and/or of geolocation, and
- Poor supplier enterprise or development hygiene.

1.2 Document Overview

This document contains the following additional sections and appendices:

Section 2 describes recommended practices customers may apply across the acquisition, deployment, and operational phases of a software supply chain.

Section 3 is a collection of appendices supplementing the preceding sections:

Appendix A: Crosswalk Between the NIST SP800-218; *Mitigating the Risk of Software Vulnerabilities by Adopting a Secure Software Development Framework* (SSDF)[4] and Use Cases described herein

Appendix B: Dependencies

Appendix C: Supply-Chain Levels for Software Artifacts (SLSA)[5]

Appendix D: Informative References

Appendix E: Acronyms.

Each section contains examples of threat scenarios and recommended mitigations**.** Threat scenarios explain how processes that compose a given phase of the software development lifecycle (SDLC) relate to common vulnerabilities that could be exploited. The recommended mitigations present controls and mitigations that could reduce the impact of the threats.

[4] Draft NIST SP 800-218, Secure Software Development Framework (SSDF) Version 1.1: Recommendations for Mitigating the Risk of Software Vulnerabilities

[5] GitHub - slsa-framework/slsa: Supply-chain Levels for Software Artifacts

2 Customer

A customer (i.e., organization) procuring and deploying a software product first defines requirements for the product and then evaluates, acquires, deploys, and maintains the product. Each of these phases comprises a series of processes which should be augmented or revised to satisfy an organization's structure and mission.

This section covers key processes in the product acquisition, deployment, operations, and maintenance phases of the software lifecycle, beginning with requirements definition and ending with software supply chain risk management operations.

It is important that the customer augment these processes based on the organization's structure, mission, and risk tolerance, as the organization implements specific mitigation activities. Mitigations that require artifacts or action from suppliers or developers are highlighted in Appendix B, *Dependences.*

2.1 Procurement and Acquisition

2.1.1 Requirements Definition

Requirements are the foundation of successful acquisitions. They are often categorized in ways predefined by internal organizations or derived from external organizations.

Functional requirements may be derived from information provided by mission execution, the enterprise, and organization-wide requirements. Non-functional requirements may include security and supply chain risk management (SCRM) activities or those activities that involve conducting gap analysis on existing solution inventories. Requirements based on defined risk profiles include inputs from information assurance data that correlates threat, vulnerability, and mitigation information; and from the Acquisition Security (ACQSEC) team, which should contain known risks about foreign ownership, control, or influence (FOCI) for the supplier landscape.

The defined risk profile may also be derived from information collected by external organizations, such as threat analytic groups. These groups provide information on known threats and risks. This information is often obtained from the U.S. Government (USG) or from commercial Cyber Threat Intelligence (CTI) providers.

Threat Scenarios

The following are example scenarios that could be exploited:

- Security requirements intended to counter threats are not domain specific or exclude organizational requirements,
- Gaps in analysis of security requirements which may lead to a mismatch of the solution or to selected security controls.

Other security requirement gaps include out-of-date security requirements or assessments and misrepresented or underrepresented risk profiles that are disconnected from acquisitions or mission areas. Insufficient security requirements may also be a problem when stakeholders are missing from signoff or approval, or when solution inventory is incomplete.

General security inadequacies may also prevail when a product isn't properly protected, when a customer is associated with suspicious geolocation and metadata, or when a customer is suspected to be associated with foreign interests.

Recommended mitigations

The following mitigations are recommended to help reduce vulnerabilities in the procurement and acquisition phase:

1. Keep security requirements and risk assessments up-to-date using business processes and require adequate protection and control of geolocation of all data and metadata.
2. Assign individual roles to:
    a) Verify the domain-specific and organizational security requirements,
    b) Coordinate risk profile definitions with mission and enterprise areas,
    c) Verify completeness of the stakeholders-list and independently review solution inventory, and
    d) For acquisitions considered either large or significant for your organization, conduct a gap analysis.
3. Additional controls for consideration include:
    a) Security requirements for all acquisitions,
    b) Original products, including any updates or upgrades, should have a hash or signature that can be independently verified by the customer,
    c) Developers and suppliers to provide the customer with guidance on how to verify the integrity of the components.
4. When acquiring software through spin-offs, external entities, or third-party suppliers, customers should:
    a) Implement continuous monitoring of the entire SCRM calculation, and
    b) Implement appropriate controls to mitigate changes to assumptions and security risks, or threats in coordination with information technology (IT) security and SCRM operations teams.

2.1.2 Product Evaluation

Product evaluation is imperative as it ensures the product complies with standards.[6] It also identifies product defects. The evaluation process, as shown in Figure 1, includes performing an internal analysis of current market and supplier solutions. Next, input from a Request For Information (RFI) or Request For Proposal (RFP) for solutions is solicited, after which that input will be assessed to determine whether requirements that incorporate security and SCRM have been

[6] A standard is an established norm or requirement for a repeatable task which is applied to a common and repeated use of rules, conditions, guidelines or characteristics for products or related processes and production methods, and related management systems practices.

met. ACQSEC evaluation, source selection and down-selection processes, and lab trials for the supplier and product selection are also part of the process.

Figure 1: Product Evaluation Process

Threat Scenario

Threat evaluation is crucial to assessing and managing risks associated with a specific product or the software supply chain in general. The following scenarios have been identified in an effort to illustrate potential threats that organizations may face during the product evaluation phase such as:

- The evaluated product is not the delivered product,
- Vulnerabilities, which were unknown at initial evaluation, are discovered later,
- It is determined that an evaluated product has had a change of ownership or control or that evaluators lack the proper expertise to assess the software under evaluation,
- External influence on the product, selection of supplier, or external labs used for product evaluations and trials, results in false or incomplete evaluation results,
- Unknown functionalities hidden within the product, which may be risky or vulnerable, when under evaluation or testing,
- Limited or no visibility into product components or requirements; visibility is needed to make informed decisions and the lack of visibility could lead to improper evaluations. Examples of this include:
    - Lack of visibility into embedded components, libraries, modules, add-ons, etc.,
    - Lack of correct, up-to-date functional, environmental, and operational requirements,
    - Lack of correct, up-to-date domain-specific & organizational security requirements and risk assessments.

Recommended mitigations

To help counter potential vulnerabilities in the software product evaluation phase, the following mitigations are recommended:

1. Verify the contents of the software bill of material (SBOM) against the product under evaluation. This verification should include attributes such as geolocation, supplier

ownership or control, Data Universal Numbering System (DUNS) verification, and past performances.

2. Subject third-party suppliers identified in the SBOM to this same evaluation to help mitigate associated risks.
3. Forge mitigations and outcome strategies as part of the SCRM determination by correlating prior notifications of vulnerabilities in supplier products with incidents presently under evaluation.
4. Continuously vet external evaluation labs and verify their independency. Require that evaluators have proper expertise in the type of software under evaluation and require an assigned function to verify that domain-specific and organizational security requirements are in place. These assigned functions should also verify the functional, environmental, and operational requirements.

2.1.3 Contracts

Customers should draft contractual agreements to enable improved software supply chain security controls and mitigations. This section outlines suggested revisions to contractual processes, identifies common threats in creating or implementing a secure software supply chain contractual mechanism, and describes recommended controls to mitigate software supply chain risks.

Customers should implement processes to create and execute contractual mechanisms requiring the appropriate level of controls for improved supply chain mitigations. Acquisition and contracts organizations should add new software supply chain-related requirements to existing and new contractual vehicles. Statements of Work (SOW) may include the software supply chain mitigations like those described in this document. Further, acquisition and contracts organizations should submit contractual artifacts to existing contractual review boards to gain necessary approvals. Once the acquisition or contracts organization receives approval for the contractual language, they should execute all resulting contracts including all necessary signatures.

Threat scenarios

The following list outlines scenarios that could yield potential software supply chain related threats to the acquisition and contracts process and result in customers experiencing a higher risk of disruption or compromise:

1. Initial execution of a contract agreement with a supplier who is under foreign control, or when the supplier changes the sourcing of products or selects a subcontractor or supplier that is under foreign control after execution.
2. Contract security requirements that are:
    a) Incomplete (e.g., lacks strong cryptographic methods for verifying product integrity), or
    b) Contract lacks SCRM requirements, as recommenced in this guide. The contract may also lack or have incomplete requirements to ensure the software supply chain integrity and security for supplier or third-party supplier.
3. An SBOM (a key SCRM requirement that supports verification of an acquired product) is missing entirely or lacks a means to ensure the integrity for the product.

4. The supplier has poor security hygiene or has experienced a compromise or other cybersecurity incident that affects the integrity of the product or the infrastructure that supports its development, thereby enabling delivery of illegitimate products, hashes and signatures.
5. The supplier alters or substitutes individual components included in the product distribution prior to package signing and hashing.

Recommended mitigations

The following mitigations are recommended to help reduce software supply chain risks in the acquisitions and contracts processes. Contracts should include terms that incorporate the following areas such as:

1. Recommended security and SCRM requirements, and terms that include new Federal Acquisition Regulation/Defense Federal Acquisition Regulation (FARS/DFARS) provisions that require visibility into the provenance of each product delivered. Terms may include:
    a. Specific security hygiene requirements for the developer and supplier as identified in item 2 below, and
    b. Requirements for suppliers to provide notification of updates and modifications to functionality, vulnerabilities, and support from time of evaluation to time of delivery.
2. Require suppliers' self-attestation of cybersecurity hygiene for their development process and the infrastructure supporting the development of their product. Ideally, self-attestation should include:
    a. New FARS/DFARS provisions for self-attestation from each supplier who provides products to USG customers,
    b. A timeline or checklist of key steps that comprise the supplier's security processes that were performed in the development of the product,
    c. Signature by a supplier-designated official responsible for the security hygiene of the development process and infrastructure,
    d. Requirement for supplier to provide cryptographic security for hashing/signature infrastructure of its product distribution system/method, and
    e. Requirement to ensure greater software supply chain visibility into the sourcing and ownership of products and suppliers.
3. Require the supplier to inform all customers on how to verify the integrity of all software components through:
    a. Verify component-level integrity by requiring the use of a hash or signature (can be same key for signing the full distribution package) or similar method to ensure the integrity of each component and require each supplier to inform the customer on how to verify the integrity of the components,
    b. Require that all artifacts sent by the supplier be in a standardized SBOM format

c. Provide SBOMs for all upgrades,

d. Ensure newly issued SBOMs incorporate all changes to the product baseline,

e. Provide continuous reporting for all of the supplier's key attributes, such as its ownership, geolocation, and foreign control as well as for any changes of the key attributes, and

f. Notify the customer of cyber incidents and investigations, mitigations, and impacts to the product or the development environment of the product. Periodicity of notifications are to include time of delivery, time period of support and maintenance.

2.2 Deployment

2.2.1 Acceptance of Product

Customers should take appropriate precautions prior to accepting the product from the supplier or developer. Upon initial receipt of the product, the customer should thoroughly examine it. This includes verifying that the product is the same as the one that was evaluated at the time of pre-acquisition evaluation, or an acceptable update, checking for evidence of tampering or substitution, and confirming receipt of required artifacts and supporting documentation for the product.

Threat scenarios

Failure to thoroughly examine the acquired product exposes the customer to several software supply chain threats. These include:

- Substitution of another product or a counterfeit product for the ordered product,
- An incomplete or incorrect version of the ordered product,
- Product tampering,
- Missing documentation, artifacts, or references for the product.

Recommended mitigations

Customers should have policies and mechanisms in place to mitigate threats from received software. Some of the actions the customer should take depend on the supplier or developer because they should inform the customer how to verify the results of the actions.

1. Verify the security of the infrastructure(s) used by the supplier for the distribution of their products. If the security is invalid or insufficient, the deployment should be blocked, and further risk response taken as needed.
2. Verify the integrity of the product, using mechanisms that include safeguards such as hashes, signatures, or certificates, as well as other out-of-band methods of verification.
3. Verify the integrity of the individual components in the distribution. This may be verified via hash or signature (possibly using the same key that signed the full distribution package), or a similar method for providing integrity of the component.
4. Verify the SBOM against the product delivered.

5. Verify the self-attestation mechanism is signed by the developer- or supplier-designated official(s) as a means of representing the security hygiene of the development process and infrastructure.

2.2.2 Functional Testing

For functional testing there should be processes for creating tests and test environments, installing and setting up of a product, executing tests, and reporting test results. The tests, and test environment, should be saved and stored for future reference and use. Final steps should include verifying the contents of the SBOM against the product under evaluation.

Threat scenario

The following scenarios could potentially yield threats:

- Product functionality unknowingly changes,
- Product contains unverified or unknown components.

Recommended mitigation

Recommended mitigations include:

- Saving and storing the tests and test environment for future reference and use,
- Verify the contents of the SBOM against the product under evaluation.

2.2.3 Assurance – Security Testing/Validation

Adversaries, whether they are nation-states or individuals, are continually looking for weakness within an organization's information system environment. These weaknesses range from misconfigured systems to lapses in the security testing or validation process. Organizations must address this ever-increasing threat vector by implementing stringent control mechanisms.

An organization should minimize threats according to their risk tolerance. To satisfy this requirement, organizations should establish and support a risk management program by understanding the risks associated with their information assets through:

- Risk identification (i.e., where and what is the risk),
- Risk analysis (i.e., how severe is the current level of risk),
- Risk evaluation (i.e., is the current level of risk acceptable).

To effectively implement the overall information assurance process through security testing or validation, as well as to keep the cyber ecosystem safe and intact, organizations should adhere to recommended mitigation strategies found in this guide as well as those provided in other resources. Such strategies include, but are not limited to, performing software risk assessments, creating software security tests, creating a software test environment, installing and setting up of software products, running software tests, validating software requirements through SCRM practices, and reporting results.

When selecting specific software security scanners, organizations should weigh the criteria for choosing a one-size-fits-all tool as such an approach may compromise the organization's overall

security posture. Effectiveness of technology, cost, presence or absence of key metrics and clear visualization, and comprehensiveness of coverage are some key factors to be considered.

Verifying whether security controls and settings that are currently in place are crucial to mitigating cyber threats within information assets. Additional layers of analysis and evaluation performed through automated penetration or security testing by red teams could help to validate whether deployed controls are effectively providing security. In addition to testing controls, red team members may use real scenarios to test actual response and detection capabilities. Blue team members also play a pivotal role by validating security controls through on-site assessments by either periodical or a continuous monitoring.

Threat scenario

Thorough and comprehensive security testing and validation is crucial because it allows organizations to prevent attacks, incidents, threats, and network compromises. Threats to information security can be in the form of software attacks (e.g. malware, file-less malware, viruses, worms, macros, denial of services, script injections, etc.) or in the form of technical software failures or errors (e.g., bugs, code problems, loopholes, back doors, etc.).

From the software supply chain threat perspective, major threats include, but are not limited to:

- Unknown changes in functionality or security of a product after deployment, complexity of the scanning, detection, and removal of malware in a virtualized environment,
- Event-driven malware where the malicious code remains dormant until a predefined date to be triggered, and product stores,
- Transferring or consuming data, metadata and content to or from external resources-both encrypted and unencrypted,
- Compromise of product development infrastructure or processes during the initial product development phase,
- Lack of proper security hygiene or cyber maturity from the supplier,
- Lack of rigorous vetting of or decreased insight into the supplier’s attributes such as ownership, geolocation, and foreign control.

Recommended mitigations

Recommended mitigation examples include:

1. Create a platform where tests and test environments can be saved for future use. This helps organizations prepare consistent test conditions and testing approaches to evaluate a product's effectiveness. Within the security test suite, include an actual runtime end-state environment to test and run the program for sufficient time to verify that no hidden or latent functionalities occur.
2. Verify data protection, implementation, and geolocation requirements match the intended customer's requirements.
3. Review and validate the self-attestation of cybersecurity hygiene of the supplier's development process and the infrastructure supporting the development of their product against security and SCRM requirements as defined by the contract in place.
4. Incorporate a continuous security validation approach by adopting a frequent testing and validation scheme in order to increase cyber resiliency.

2.2.4 Configuration Control Board Review

Product operations, maintenance, and support is a key part of the software product's lifecycle. Configuration and change management are central to IT service management (ITSM) and should be overseen by an organization's Configuration Control Board (CCB) also known as the Change Control Board. Front-end responsibilities of the CCB with respect to software include:

- Receiving functional and assurance findings such as reports and testing results,
- Determining risk of the software, both on its own and in terms of interdependencies with existing systems and software,
- Determining any changes needed for authorization to operate (ATO) or equivalent internal approvals,
- Making a go/no-go decision on the product,
- If applicable, documenting the approved configuration of the product with all future changes to product configuration documented and that documentation maintained.

Threat scenarios

The following scenarios are indicators of vulnerabilities in the CCB review which may be exploited:

- The CCB receives inaccurate or incomplete product reports from developers, suppliers, testers, etc., which limits their ability to make accurate decisions,
- The CCB receives insufficient information on the product's functionality, operations, intended use, etc.,
- CCB members lack specific knowledge or expertise on the product or the product's overall technology area to make an informed decision,
- The CCB review is biased by internal or external factors, leading to less-than-optimal decisions.

Recommended mitigations

The customer may use the following mitigations to minimize the likelihood of these threats within the CCB review process from occurring:

1. Evaluate organizational SCRM requirements for all supporting product documentation from developers, suppliers, testers, etc.
2. Use protected and secure channels of communication to relay functional and assurance reports and product testing results.
3. Ensure subject matter experts are on the CCB or that they are consulting with the CCB.
4. Validate and document the CCB decision-making process as it proceeds.

2.2.5 Integration

After procurement, the customer's IT team puts the product into operation for evaluation. Next, the team plans for integration and roll-out. This involves establishing integration processes and implementing or configuring controls for threats that are common to the integration phase.

The integration process begins with establishing an integration plan. Integrators will need to leverage both the functional requirements as well as the functional test phase data (i.e., configuration, environment, and results data) to define the integration plan. The plan will need to account for the integration of the software or product into its operating environment (enterprise, cloud, hybrid, client, etc.) and the deployment of configuration controls specific to the software or product within its operating environment.

Product modification is integral throughout the integration process. Internal offices and third-party suppliers may need to modify and adjust development of the product which would not only support the actual integration but also support functionality to meet customer needs. Additionally, any product modifications will need to be developed and tested.

Exploited software supply chain vulnerabilities may open the door to additional vulnerabilities. Assumptions built into the product may be violated during testing that could lead to potentially vulnerable conditions. Also, either wittingly or unwittingly, internal development may introduce software supply chain vulnerabilities such as compromised components, backdoors, or unknown documents, functions or features.

Threat scenarios

Testing is a key part of product integration. The software or product will need to undergo functional testing, interoperability testing, and production security testing. The latter leverages assurance testing, validation plan tests, and environmental configuration. An organization's internal security processes may also introduce exploitable vulnerabilities.

The integration process requires reviewing the product for vulnerabilities that are common to integration, such as the following:

- Undocumented software modifications or a supplier's inability to support the product after customer modifications,

- Products that hide malicious functionality during functional, interoperability, and security testing or products that exfiltrate trust artifacts provided to it (e.g., CodeCov[7] exfiltrating credentials),
- Internal product modifications where integrators and testers use their own tools or monitoring capabilities rather than the enterprise tools under which the product will be deployed.

Recommended mitigations

The following mitigations may be used to minimize the likelihood of threats occurring during the integration process:

1. Due to the threat that malicious actors compromised the test environment, use multi-layered defenses (i.e., Zero Trust).
2. During testing, utilize test credentials and other artifacts where possible to avoid giving real IDs.
3. Make the integration test environment realistic by using tools that reflect the deployment environment.
4. Continuously monitor the test environment and product during operation to augment integration testing.
5. Block beaconing to malicious and unneeded sites.
6. Minimize trust artifacts given to products.
7. Maintain and review logs for evidence of unexpected behavior.
8. Conduct updated security testing to include any post-acquisition product modifications.
9. Where possible, coordinate product modifications with the supplier or seek contract support from the third party that performed the modifications and ensure documentation for product.
10. Add the software or product to the application allowlist in addition to any security and network monitoring lists or tools.
11. Deploy security controls for the software or product in its operating environment.

An important process for product integration is establishing trust relationships for the new software or product within the organization and partner-organizations. This includes assigning administration rights and rights for credentials, permissions, accesses, applied people, devices, and entities. The integration process requires creation of rules, roles, and groups to integrate the new product within the existing security infrastructure.

2.2.6 Roll-Out of Initial Product

After the software product is integrated into the environment, the deployment phase begins. Environments into which products may be installed and deployed includes client systems, enterprise network software installations, cloud service offerings, and hybrid deployments. After

[7] CodeCov is a tool that is used to measure the test coverage of a codebase. It generally calculates the coverage ratio by examining which lines of code were executed while running the unit tests.

the software is installed, the deployment team validates the success of the installation and integration into the environment and typically performs and documents a post-installation analysis.

Threat scenarios

Examples of vulnerabilities which create potential threats and risks during the deployment process include:

- The software product deployed is an altered version from intended configuration or condition,
- The team that performed the deployment does not complete a readiness assessment prior to deployment or before the deployed product goes live,
- The product activates malicious functionality after time has elapsed, or when a condition is satisfied,
- The software product disables defensive measures prior to—or as part of—activation of malicious functionality, often in a covert way that is hard to detect, such as the malware inserted into the SolarWinds software,
- The software product falsifies results to validate assessments,
- The software developer intentionally or unintentionally leaves backdoors in the product,
- The organization has poor cybersecurity posture or cyber hygiene in general, leaving threats unmitigated.

Recommended mitigations

The following mitigations can help reduce threats and risks associated with the deployment process:

1. The developer or supplier should inform the customer on how to verify the integrity mechanisms so that the deployment team or supporting IT function can check and verify the software product's integrity mechanisms (e.g., product file hashes, config file hashes) during deployment to ensure integrity.
2. Deployment teams should not rely solely on product self-reporting and should employ an independent means of validation.
3. After validation, the software product can be added to the application allowlist for the given environment.
4. As part of Security Operations, continuously monitor the product, including for unauthorized encrypted sessions that can't be inspected, and block these where possible.
5. The security team should (in collaboration with the deployment team) implement strict least privilege for product access and resources using default deny policies where possible.
6. As part of Security Operations, product-specific monitoring should be performed based on risk management determination for criticality.

2.2.7 Upgrade of Product

During the lifecycle of a software product, a supplier may develop and deploy a newer version to the customer. When the customer's IT organization creates and implements a product upgrade plan, the plan should include performing a set of integration, security, and interoperability tests. The scope of these tests should be based on the customer's risk tolerance for the upgrade. The plan should identify and incorporate tests for new functionality, or changes in security assumptions being introduced by the upgrade. The test plan should include regression testing that incorporates previous test results from prior upgrades and the original product deployment. Application allowlists should be updated to allow for the upgraded version of the software product. Finally, IT should assign or change any new permissions, credentials, rights, policies, or roles as required by the software product's upgrade.

Threat scenarios

Throughout the processes required to implement and deploy software upgrades, vulnerabilities may exist or be introduced, exposing the software to certain threats. Examples of vulnerabilities which create potential threats and risks during the upgrade process include:

- The team performing the upgrade lacks a thorough readiness assessment and plan prior to deployment of the upgrade or before the upgrade goes live,
- Software upgrades are compromised or altered from the intended configuration or condition,
- Product upgrades falsify testing/validation results or introduce malicious functionality, which activate after some time has elapsed or a condition has been satisfied,
- The functionality of an upgrade disables defensive measures prior to or as part of a malicious functionality being activated (often in ways that are hard to detect, e.g., the malware that compromised SolarWinds),
- Product developers intentionally or unintentionally introduce backdoors in the product that lead to exposures.

A lack of readiness assessment and planning could lead to a lack of appropriate controls and validation of the upgrade to the software product. It could also lead to substitution of products via the upgrade, missing documentation and artifacts, and wrong versions or incomplete upgrades to the product. A more general risk may exist if the organization has a poor cybersecurity posture, resulting in many of these risks going unmitigated.

Recommended mitigations

The following recommendations for security controls and mitigations could help reduce the risks and threats associated with the deployment process of an upgrade:

1. The deployment team or a supporting IT function should verify the veracity of the software product upgrades and integrity mechanisms (e.g., product file hashes, configuration file hashes, and component hashes) to ensure integrity.
2. The upgrade deployment team should verify the integrity of the method and infrastructure from where the upgrade was obtained (e.g., Website, CDN, push mechanism, etc.).

3. The upgrade deployment team should verify the updated SBOM against the product delivered and confirm that the self-attestation is signed by a supplier designated official responsible for the security hygiene of the development process or infrastructure.
4. The upgrade deployment team should employ independent means of software validation, not solely relying on product self-reporting.
5. After validation, the upgraded software product should be added to the application allow list for the given environment.
6. The security operations team should continuously monitor for unauthorized encrypted sessions that can't be inspected and consider blocking encrypted sessions if possible.
7. The security operations team should perform product-specific monitoring based on risk management determination for criticality.
8. The upgrade deployment team (in collaboration with the security operations team) should implement strict least privileges for product access and resources using default deny policies where possible.
9. The security operations team (in collaboration with the upgrade deployment team) should compare behavior pre- and post-upgrade and investigate any differences.

2.2.8 Product End-of-Life

As part of the lifecycle of any product, the customer organization may decide to remove the software product from its systems. This could be due to a decision to move to a different software product supplier, end of need or mission, end-of-life (EoL) of product by supplier or developer, a risk management determination, or other reasons. The customer IT organization should assign a team to plan, manage, and implement EoL procedures and removal of the software product from the organization. The plan should include the process for removing the software, provisions for exceptions, and a training plan (including content development and training for pre and post removal). The team should coordinate with IT security operations to remove all trust relationships associated with the software product as well as to create a security monitoring plan for the period of the removal and for exceptions to the removal.

The security operations team should document and implement a plan for removal of trust relationships. The plan should include:

- Deactivation of credentials,
- Deactivation of permissions/accesses/rights (applied to people, devices and entities),
- Deactivation of the product administrator accounts,
- Removal of all rules/roles/groups for product in existing security infrastructure, (except for what is implemented as exceptions per the exception plan),
- Removal of the product from application allowlists,
- Adding the software product to the denylist in security and network monitoring tools.

Security operations should use segmentation and isolation security procedures for exceptions to the removal plan.

Threat scenarios

Even with detailed plans and procedures in place, a few threats may persist within the end-of-life or software product removal process:

- The product leaves some functionality behind, causing an exposure or unmonitored risks,
- Credentials or other trust artifacts are not revoked or removed, leaving an exposure,
- An inexperienced team is assigned to the EoL procedures and plan development, leading to risks and exposures to the process and post process environment,
- Dependency or reliance on other software products, workflows, or on other systems for the product that is being removed, leading to exposure or disruption of those other systems.

Recommended mitigations

1. The customer IT team assigned to the EoL process should collaborate with the security operations team to:
    a. Create and implement a plan to remove all trust relationships associated with the software product,
    b. Create a security monitoring plan for the period of the removal as well as for exceptions to the removal.
2. The security operations team should document and implement a plan for the removal of trust relationships. This plan should include:
    a. Deactivation of credentials/permissions/accesses/rights-applied to people, devices and entities,
    b. Deactivation of the product admin accounts,
    c. Removal of all rules/roles/groups for a product in an existing security infrastructure, except for what is implemented for exceptions per the exception plan, and
    d. Removal of a product(s) from application allowlists, and adding the software product to the denylist in security and network monitoring tools.
3. The security operations team should use segmentation and isolation security procedures for exceptions to the removal plan.
4. The customer IT team assigned to the EoL process should have a standard operating procedure for decommissioning software products that includes checking for residual functions, files, etc. The team should use an asset management tool to verify what is in the environment and validate that the software product being decommissioned or removed has actually been removed and decommissioned.
5. The IT team and the security operations team should use and implement a testing plan to validate that trust artifacts for the product have been fully decommissioned. A plan should be jointly created and implemented between the customer IT team assigned to the EoL and the security operations team to handle exceptions to the software removal.
6. The customer IT team assigned to EoL activities should perform a dependency analysis prior to EoL to identify software, workflow, and systems dependencies that may adversely

affect the organization and its processes after the software is removed. An independent verification of the software product removal should then be implemented (i.e. red team v. blue team checks).

2.2.9 Training/Enablement

A key element that ensures successful and safe deployment of software products is having an established and effective training program in place for new software products. Training on new software should be created and rolled out as a comprehensive activity across an organization, with content tailored for network administrators, security administrators, and any key user of the product. Best practices also dictate that key users receive regular ongoing product training on a cadence appropriate to the sensitivity and risk associated with the product.

Threat scenarios

For organizational efficiency, training is usually rolled out through the use of training software that is installed on the operational system. This form of dissemination exposes the training to some degree of threat. But even when training is provided in-person, associated risk may still risks arise:

- Training software may be compromised by an adversary, resulting in disruption of training availability or changing of instructional content,
- Training systems or their dependencies can be compromised to send back information to a malicious actor, such as training content or trainee information,
- Training may omit key elements of how to securely operate the software.

Recommended mitigations

The following controls should be adopted by the customer to prevent software training from adding to potential threats and risks:

1. Implement mechanisms (administrative, policy and technical) to ensure the integrity of training software and/or systems.
2. Discourage and filter unnecessary communications of training and training software outside of the training environment or organization.
3. Apply good security hygiene on the training environment.
4. Develop a checklist of key product security features and risks and ensure these items are included in the product training.

2.3 Software operations

2.3.1 User

The highly distributed nature of computing environments requires the end-user or other responsible parties to contribute towards the goal of maintaining the IT security of the enterprise. Promptly reporting bugs or other anomalies in their use of IT systems and applications is a primary way a user may contribute. While this may increase volumes for the helpdesk, some of the volume will be quickly remedied through direct education. However, it is important to encourage users to

promptly report anything they see as anomalous, as most users do not have the training to discern the difference between a mistake and a potential security issue. User reports that aren't clearly addressable by on-the-spot user education, should be investigated and resolved or escalated to the security operations center (SOC) for further analysis.

Threat scenarios

User reports of anomalous behavior can indicate potential or active security exposures such as:

1. Reports explained as the result of a software update or functionality change.
2. The identification of bugs and possible security vulnerabilities, but some users may not be experienced in such analyses.
3. Trends and correlations with other activities may only be identified through aggregation,
4. Patches may already be available to address the issue. Therefore, all other cases should be logged, escalated to the supplier, and shared with other customers.
5. Some software may be configured to accept updates without requiring user interaction or knowledge, in other cases the users themselves may have initiated the software update.

Recommended mitigations

Customers should have policies and mechanisms in place to carry out activities that can mitigate threats encountered through software received or purchased as follows:

1. All users, regardless of responsibility or experience, should be trained on how to detect and report anomalies. In some situations, a "bug bounty" program may be helpful to encourage both users and customers to report anomalies.
2. All reports to the helpdesk should be logged regardless of immediately perceived importance.
3. All users should understand when it is appropriate to apply updates to a system.
4. Enterprises should be able to disable, or at least isolate, a particular product. Users instructed to do so may elect not to do so to complete their work or mission.
5. Enterprises should have a mechanism for informing users of product anomalies and, when potential security concerns exist, how to disable that product. The enterprise may be better served if this software disabling capability could be accomplished through a central point.
6. Enterprises should have a formal process for notifying a supplier of these anomalies and tracking these reports until resolved and distributed to the users and customers. If such reports go unresolved for an unreasonable period of time the enterprise should seek alternatives to that product.

2.3.2 Updates

Updating products is necessary and poses some risk. Following an update, the customer may perceive the product as operating differently, it may not interoperate with other products and services as before, or it may not work at all. Some updates may pose a higher risk than others, such as operating system updates, critical system application updates, and updates to customer-facing

systems. Updates should be applied only through a defined and monitored fashion. An enterprise should have trusted processes in place whereby the customer and supplier may exchange information regarding updates. These processes should include notifications that an update or patch is available, how it can be obtained or distributed (if necessary), and a way to verify the authenticity and integrity of the update or patch once it is received by the enterprise. In the case of end-user applied updates or patches, the customer must have a similar trusted process for verifying the authenticity and integrity of an update or patch.

Threat scenarios

Processes should be in place to help mitigate this risk, subject to the organization's tolerance of risk relating to that product's role. Recognize that updates can present potential risks to the product as depicted in the following scenarios:

1. Updates coming from unreliable sources (e.g., a supplier or provider other than a company's usual software supplier or provider) with no clear means of verifying its origin. There must also be a trusted means of verifying the integrity of the update to protect against software supply chain interference.
2. New capabilities or functionalities revealed during integration, interoperability, and security regression tests interfere with existing operations. Any adverse interactions or security problems should be promptly reported to the provider and a risk assessment performed to determine whether to deploy the update.
3. Undocumented or unclear instructions on how a customer should implement an update, to include updating application allowlists, access control policies, credentials, or other roles; and reverifying the integrity of the update on the customer system.
4. The software supply chain itself could be unsecure, resulting in an update that has been tampered with or replaced. Personnel or customers might apply the update incorrectly or fail to include the associated updates to access control, etc., as mentioned above. When updates are applied by customers, care must be taken to ensure that the update comes from a trusted source when it arrives on their system. Otherwise, an insider might attempt an internal software supply chain attack.

Recommended mitigations

Updating systems is a necessary part of IT management. As such, it is an attractive target for adversaries. Tampering with updates, replacing them with false updates, valid updates that disrupt normal system functionality (by accident or by design), or outright disruption of the flow of updates are all possibilities. To address the risks posed by updates, several controls should be implemented:

1. Ensure updates are only retrieved from authorized and authenticated sources via approved channels. The developer or supplier should provide detailed update notes with all changes documented. The developer or supplier should also inform the customer on how to verify the integrity of the infrastructure and mechanism provided by the update. This could take the form of hashes, signatures, or other cryptographically sound mechanisms, or via out-of-band measures.

2. Ensure the integrity of the updated SBOM and the contents of the update using automated analyses. Similar means must be applied in the case of updates containing other updates or updates to multiple subsystems or applications.
3. Perform automated security and functionality tests on updates to software, including comparisons of system behavior pre and post update. In the case of unexpected and/or undocumented differences, suspend the deployment of the update until these discrepancies are resolved.
4. Implement continuous monitoring of update sources and methods.
5. Implement continuous monitoring of software versions across the enterprise, including end-user systems and remote systems and sensors.

2.3.3 Security/Supply Chain Risk Management Operations

Supply Chain Risk Management (SCRM) operations are crucial to systematically assess potential risks, increase supplier visibility, and reduce the overall software supply chain driven impact of an organization. To this end, workflows have been categorized into various processes, threats, and recommended controls accordingly.

Defining and maintaining a master baseline for new software or products through its host or device name, network, infrastructure (cloud), etc. is critical to keep the overall information system boundary safe and intact. Continuous monitoring of the product for changes in functionality or security events, as well as incorporating a comprehensive vulnerability management program and best practices are highly recommended. In addition to conducting ongoing credentials and rights management, and putting an incident reporting and response initiative in place, organizations should determine whether a network is running optimally through network monitoring, including Domain Name Systems (DNS) monitoring, which ensures maximum health. Organizations should perform regular security testing, audits of software or products, and audits of their environments or infrastructures. Finally, implementation of continuous monitoring of the entire SCRM calculation with the ability to implement appropriate controls to mitigate changes, to assumptions and security, or risks and threats in coordination with the procurement team is highly recommended.

Threat scenarios

Software supply chain driven risks (internal or external) are a serious threat and can cause disruptions to operations and processes such as:

- Malicious software that disables, negates or hides from security agents or monitoring tools in place in user environment,
- Appropriate logs not being collected, analyzed, or correlated; and partial/incomplete continuous monitoring and security audit(s).

Recommended mitigations

Organizations should control and monitor their own risk posture as well as software supply chain driven risks from immediate suppliers and third-parties. To keep data ecosystems safe, organizations should implement recommended SCRM controls to mitigate both known and unknown software supply chain related risks. Some recommended practices are:

1. Enabling integrity protections on all deployed security agents.
2. Deploying out-of-band security tools to monitor from off-system (trust boundaries).
3. Regular and continuous red team (threat) hunting and security event exercising.
4. Risk management-based approaches to identify logging and event monitoring for specific software products.
5. Security operations training on monitoring and security analysis of software products.
6. Implementing mechanisms to rapidly instrument and deploy sensors to software product specific events and Indicators of Compromise (IOCs)—which includes projection of future capabilities to increase fidelity of response to data collected from sensors (e.g., AI /ML).
7. Implementing a threat model based on the software product's specific risks and threats and updating the threat model with latest adversary and product risk intelligence or factors.
8. Performing regular threat modeling on software products within the current system environment and leveraging cross government and industry threat sharing mechanisms to address software supply chain risks.

3 Appendices

3.1 Appendix A: Crosswalk between Scenarios and SSDF

The section reference numbers in the below crosswalk may look similar for each role (Developer, Supplier and Customer) however they are from the respective parts of the Series. (PO – Prepare Organization; PW - Produce Well-Secured Software; PS – Protect Software; and RV – Respond to Vulnerabilities)

SSDF #	Developer	Supplier	Customer
PO.1	2.2.3 Secure Development Practices	2.1.1 Define criteria for software security checks	
PS.1	2.2.1.1 Source Control Check-in Process 2.2.1.4 Code Reviews 2.2.6 External Development Extensions 2.3.2 Selections and Integration 2.4.1 Build Chain Exploits 2.5.3 Secure the Distribution System	2.2.1 Protect all forms of code from unauthorized access 2.2.2 Provide a mechanism for verifying software release integrity (PS.1, PW.9)	
PS.3	2.2.1.1 Source Control Check-in Process 2.2.1.2 Automatic and Manual Dynamic and Static Security / Vulnerability Scanning 2.3.2 Selections and Integration 2.3.3 Obtain Components from a Known and Trusted Supplier 2.4.1 Build Chain Exploits	2.2.3 Archive and protect each software release	
PW.1	2.3.2 Selections and Integration	2.3.1 Design software to meet security requirements	
PW.3	2.2.3 Secure Development Practices 2.3.2 Selections and Integration	2.3.2 Verify third-party software complies with security requirements	2.1 Procurement/Acquisition (1) Requirements Definition / Recommended Controls (viii)(viii)

	2.3.3 Obtain Components from a Known and Trusted Supplier 2.3.4 Component Maintenance 2.3.5 Software Bill of Material (SBOM)		2.2 Deployment (6) (2) Testing – Functionality (c) Recommended Controls (ii) Verify contents in SBOM 2.2 Deployment (6) Deploy (3) Contracting / Recommended Controls (v) (viii) (ix)(x)
PW.6	2.2.3.2 Use of Unsecure Development Build Configurations 2.4.1 Build Chain Exploits	2.3.3 Configure the compilation and build processes	
PW.7	2.2.1.4 Code Reviews 2.2 Open-Source Management Practices 2.2.6 External Development Extensions 2.3.2 Selections and Integration 2.3.3 Obtain Components from a Known and Trusted Supplier	2.3.4 Review and/or analyze human-readable code	
PW.8	2.2.1.3 Nightly Builds with Regression Test Automation 2.3.2 Selections and Integration 2.4.1 Build Chain Exploits	2.3.5 Test executable code	
PW.9	2.2.3.2 Use of Unsecure Development Build Configurations 2.4.1 Build Chain Exploits	2.2.2 Provide a mechanism for verifying software release integrity (PS.1, PW.9) 2.3.6 Configure the software to have secure settings by default	
RV.1	2.3.4 Component Maintenance 2.4.1 Build Chain Exploits	2.4.1 Identify, analyze, and remediate vulnerabilities on a continuous basis	

3.2 Appendix B: Dependencies

3.2.1 Customer Group Dependencies

Dependencies and artifacts recommended to be performed or provided by the supplier for benefit of the customer to aid in adequately evaluating and assessing software.

#	Dependency
1	SBOM (or similar Software Bill of Materials artifact) for product
2	SBOM (or similar Software Bill of Materials artifact) for product update
3	SBOM (or similar Software Bill of Materials artifact) for product upgrade
4	Verifiable integrity (e.g., hash/signature) of product distribution package
5	Verifiable integrity (e.g., hash/signature) of product update
6	Verifiable integrity (e.g., hash/signature) of product upgrade
7	Verifiable integrity (e.g., hash/signature) of suppliers' product distribution system/method infrastructure
8	Verifiable integrity (e.g., hash/signature) of product components in distribution package
9	Self-attestation artifact of cybersecurity hygiene of their development process and the infrastructure supporting the development of their product (see artifacts appendix)
10	The self-attestation is signed by responsible official or executive
11	Notification from supplier of all updates and modifications to functionality, vulnerabilities, and support from time of evaluation to time of delivery of product
12	Artifacts provided by supplier that document attributes such as geo-location, supplier ownership/control, DUNS (if applicable), past performance
13	Artifacts provided by supplier that document attributes such as the available geo-location, supplier ownership/control, DUNS, past performance for third-party suppliers identified in the SBOM
14	Artifacts sent by supplier will be in a standardized format (e.g., SBOM)
15	Notification from supplier of changes to/of supplier ownership, geo-location, and control of supplier, supplier, and third parties
16	Notification from supplier of cyber incidents and investigations and mitigations and any impacts to the product or the development environment of the product at time of delivery
17	Notification from supplier of cyber incidents, investigations and mitigations, and any impacts to the product or the development environment of the product post acquisition (for period of support and maintenance)

3.2.2 Developer Group Dependencies

Green – Dependencies and artifacts recommended to be provided by the supplier for benefit of the developer.

Dark Green – Dependencies and artifacts recommended to be provided by the third-Party suppliers for benefit of the developer.

Pink – Dependencies and artifacts recommended to be provided by the customer for benefit of the supplier/developer.

#	Dependency
1	Provide issues from customers
2	Provide given hashes as required
3	SDLC policies and procedures
4	Secure architecture, high-level design
5	Qualified team assembly with code/security training
6	Independent QA individual/team
7	Independent security audit individual/team
8	Open-Source Review Board (OSRB) with repository
9	Product release management/resources
10	SBOM
11	Development location and information
12	Third-party SBOM
13	Third-party License
14	Release notes (detailing vulnerabilities fixed)
15	Vulnerability notifications
16	Publish updates and patches to the customer to address new vulnerabilities or weaknesses found within the product
17	Requirements and criteria for success
18	Implied industry security requirements
19	Provide issues from operational environment, take updates and patches
20	Vulnerability notifications and reporting from the users

3.3 Appendix C: Supply-Chain Levels for Software Artifacts (SLSA)

Supply-Chain Levels for Software Artifacts (SLSA) is a security framework from source to service, giving anyone working with software a common language for increasing levels of software security. The framework is currently in Alpha stage and constantly being improved by supplier-neutral community. Google has been using an internal version of SLSA since 2013 and requires it for all of their production workloads. http://slsa.dev

Requirement	Description	L1	L2	L3	L4
Scripted build	All build steps were fully defined in a "build script." The only manual command, if any, was to invoke the build script. Examples: • Build script is Makefile, invoked via make all. • Build script is. github / workflows / build.yaml, invoked by GitHub Actions.	✓	✓	✓	✓
Build service	All build steps run using a build service, not on a developer's workstation. Examples: GitHub Actions, Google Cloud Build, Travis CI.		✓	✓	✓
Ephemeral environment	The build service ensured that the build steps ran in an ephemeral environment, such as a container or VM, provisioned solely for this build, and not reused from a prior build.			✓	✓
Isolated	The build service ensured that the build steps ran in an isolated environment free of influence from other build instances, whether prior or concurrent. • It MUST NOT be possible for a build to access any secrets of the build service, such as the provenance signing key. • It MUST NOT be possible for two builds that overlap in time to influence one another. • It MUST NOT be possible for one build to persist or influence the build environment of a subsequent build. • Build caches, if used, MUST be purely content-addressable to prevent tampering.			✓	✓
Parameterless	The build output cannot be affected by user parameters other than the build entry point and the top-level source location. In other words, the build is fully defined through the build script and nothing else. Examples: • GitHub Actions workflow dispatch inputs MUST be empty.				✓

	• Google Cloud Build user-defined substitutions MUST be empty. (Default substitutions, whose values are defined by the server, are acceptable.)				

Hermetic	All transitive build steps, sources, and dependencies were fully declared up front with immutable references, and the build steps ran with no network access.
	The developer-defined build script:
	• MUST declare all dependencies, including sources and other build steps, using immutable references in a format that the build service understands.
	The build service:
	• MUST fetch all artifacts in a trusted control plane.
	• MUST NOT allow mutable references.
	• MUST verify the integrity of each artifact.
	  ○ If the immutable reference includes a cryptographic hash, the service MUST verify the hash and reject the fetch if the verification fails.
	  ○ Otherwise, the service MUST fetch the artifact over a channel that ensures transport integrity, such as TLS or code signing.
	• MUST prevent network access while running the build steps.
	  ○ This requirement is "best effort." It SHOULD deter a reasonable team from having a non-hermetic build, but it need not stop a determined adversary. For example, using a container to prevent network access is sufficient.
					✓

Reproducible	Re-running the build steps with identical input artifacts results in bit-for-bit identical output. Builds that cannot meet this MUST provide a justification why the build cannot be made reproducible. "○" means that this requirement is "best effort." The developer-provided build script SHOULD declare whether the build is intended to be reproducible or a justification why not. The build service MAY blindly propagate this intent without verifying reproducibility. A customer MAY reject the build if it does not reproduce.				○

3.4 Appendix D: Informative References

In principle, any artifacts created during the lifecycle of the software development process are owned by and private to a developing organization. These organizations can determine what artifacts are made available with potential and current customers of a product with or without a Non-Disclosure Agreement (NDA). Availability of information must take into consideration regulatory and legal requirements, the customer requirements for the information and the risk involved by exposing information leading to the exploitation of the product. Exceptions may include open source development organizations, which are more inclined to make all development information available, to include source code.

When defining the availability of an artifact, the general terms used in this section will be the following:

1. Publicly disclosed,
2. Externally available,
    a) under a Non-Disclosure Agreement (NDA),
    b) government agency mandated requirement,
3. Private / company confidential.

The availability of an artifact varies between companies and agencies and is only described here as a reference for what might be possible when using artifacts to validate the software supply chain process. Some artifacts, such as a high-level architecture document may be intentionally generated to allow any perspective consumers an introductory artifact detailing the overall strategies used in the design, development, and operation of a product. These publicly disclosed documents may describe common industry nomenclature, such as Federal Information Process Standards (FIPS) compliance, cryptography standards used, development processes adhered to or certifications processes passed. NDA and government mandated availability require contractual agreements providing access to artifacts that would not normally be exposed by the organization that produced the product. While private/company confidential artifacts are generally low-level and detailed work products that may contain sensitive secrets and knowhow and if exposed, provide potential insight into product's competitive implementation and threat vectors that may not be addressed in the product, therefor posing a threat if exposed outside of the producer's environment. Private/company confidential artifacts are generally maintained by the "Suppliers" and "Developers" of the product to facilitate the auditing and validation of adherence to the Secure Software Development Lifecycle (Secure SDLC) and security practices set forth by the product owner, company, or organization. For more information on the Secure SDLC process, refer to Section **2.1 "Secure Product Criteria and Management"**, subsection Recommended Mitigations, Item 8 of Part 1 Developers of the series.

Most of the artifacts collected during the development lifecycle are not meant to be shared outside the developing organization yet may be preserved in persistent storage as evidence to verify the integrity of the policies and processes used during the development of a product. A developer should securely retain artifacts of software development for a certain duration according to the secure software development policies and processes. As a by-product of the process used to implement and mitigate the attack surface and threat model of the software as well as the software build pipeline during the development process, the following artifacts may be created, and collected:

Abbreviation	Document Name
ACM	Communications of the ACM 17, "The Protection of Information in Computer Systems". Available at (http://web.mit.edu/Saltzer/www/publications/protection/index.html)
BSA	BSA (2019) Framework for Secure Software. Available at (https://www.bsa.org/reports/bsa-framework-for-secure-software)
BSIMM10	Migues S, Steven J, Ware M (2019) Building Security in Maturity Model (BSIMM) Version 10. Available at (https://www.bsimm.com/download/)
CISA	Cybersecurity & Infrastructure Security Agency. Available at (https://www.cisa.gov/defining-insider-threats)
CISCO_SDLC	Cisco. 2021. Cisco Secure Development Lifecycle. Available at (https://www.cisco.com/c/dam/en_us/about/doing_business/trust-center/docs/cisco-secure-development-lifecycle.pdf)
EO14028	EOP. 2021. "Improving the Nation's Cybersecurity", Executive Order 14028, 86 FR 26633, Document number 2021- 10460. Available at (https://www.whitehouse.gov/briefing-room/presidential-actions/2021/05/12/executive-order-on-improving-the-nations-cybersecurity/)
FIPS140	National Institute of Standards and Technology. 2019. "Security Requirements for Cryptographic Modules." Available at (https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.140-3.pdf).
IDASOAR	Hong Fong EK, Wheeler D, Henninger A (2016) State-of-the-Art Resources (SOAR) for Software Vulnerability Detection, Test, and Evaluation 2016. (Institute for Defense Analyses [IDA], Alexandria, VA), IDA Paper P-8005. Available at (https://www.ida.org/research-and-publications/publications/all/s/st/stateoftheartresources-soar-for-software-vulnerability-detection-test-and-evaluation-2016)
INTEL	Intel. Software Supply Chain Threats; A White Paper
ISO27034	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, 2011. Available at (https://www.iso.org/standard/44378.html)
MITRE_CAPEC	MITRE. 2021. Common Attack Pattern Enumeration and Classification. Available at (https://capec.mitre.org/data/definitions/437.html)
MITRE_CVE	MITRE. 2021. "Common Vulnerability and Exposure, CVE." 2021. Available at (https://cve.mitre.org/index.html).
MSSDL	Microsoft (2019) Security Development Lifecycle. Available at https://www.microsoft.com/en-us/sdl
NASASTD8739	National Aeronautics and Space Administration. 2021. "SOFTWARE ASSURANCE AND SOFTWARE SAFETY STANDARD, NASA-STD-8739.8A." Available at

	(https://standards.nasa.gov/sites/default/files/standards/NASA/PUBLISHED/A1/nasa-std-8739.8a.pdf).
NICCS	National Initiative for Cybersecurity Careers and Studies, National Initiative for Cybersecurity Education. 2021 Workforce Framework for Cybersecurity (NICE Framework). Available at (https://niccs.cisa.gov/workforce-development/cyber-security-workforce-framework)
NISTCSF	National Institute of Standards and Technology. 2018. "Framework for Improving Critical Infrastructure Cybersecurity, Version 1.1." Available at (https://doi.org/10.6028/NIST.CSWP.04162018)
NISTMSDV	National Institute of Standards and Technology. 2018. "Guidelines on Minimum Standards for Developer Verification of Software". available at (https://www.nist.gov/system/files/documents/2021/07/13/Developer%20Verification%20of%20Software.pdf)
NTIASBOM	National Telecommunications and Information Administration. 2021. "The Minimum Elements for a Software Bill of Materials (SBOM)." Available at (https://www.ntia.doc.gov/report/2021/minimum-elements-software-bill-materials-sbom)
NVD	National Vulnerability Database. Available at (https://www.nist.gov/programs-projects/national-vulnerability-database-nvd)
OWASP_ASVS	Open Web Application Security Project (2019) OWASP Application Security Verification Standard 4.0. Available at https://github.com/OWASP/ASVS
OWASP_SAMM	Open Web Application Security Project (2017) Software Assurance Maturity Model Version 1.5. Available at (https://www.owasp.org/index.php/OWASP_SAMM_Project)
OWASP_SCVS	OWASP. 2021. "OWASP Software Component Verification Standard." Retrieved Sep. 25, 2021 (https://owasp.org/www-project-software-component-verification-standard/).
OWASP_TEST	Open Web Application Security Project (2014) OWASP Testing Guide 4.0. Available at https://www.owasp.org/images/1/19/OTGv4.pdf
PCI_SSLRAP	Payment Card Industry (PCI) Security Standards Council (2019) Secure Software Lifecycle (Secure SLC) Requirements and Assessment Procedures Version 1.0. Available at (https://www.pcisecuritystandards.org/document_library?category=sware_sec#results)
SC_AGILE	Software Assurance Forum for Excellence in Code (2012) Practical Security Stories and Security Tasks for Agile Development Environments. Available at (http://www.safecode.org/publication/SAFECode_Agile_Dev_Security0712.pdf)
SC_FPSSD	Software Assurance Forum for Excellence in Code (2018) Fundamental Practices for Secure Software Development: Essential Elements of a Secure Development Lifecycle Program, Third Edition. Available at (https://safecode.org/wpcontent/uploads/2018/03/SAFECode_Fundamental_Practices_for_Secure_Software_Development_March_2018.pdf)

SC_SIC	Software Assurance Forum for Excellence in Code (2010) Software Integrity Controls: An Assurance-Based Approach to Minimizing Risks in the Software Supply Chain. Available at (http://www.safecode.org/publication/SAFECode_Software_Integrity_Controls0610.pdf)
SC_TPC	Software Assurance Forum for Excellence in Code (2017) Managing Security Risks Inherent in the Use of Third-Party Components. Available at (https://www.safecode.org/wpcontent/uploads/2017/05/SAFECode_TPC_Whitepaper.pdf)
SC_TTM	Software Assurance Forum for Excellence in Code (2017) Tactical Threat Modeling. Available at (https://www.safecode.org/wpcontent/uploads/2017/05/SAFECode_TM_Whitepaper.pdf)
SLSA	The Linux Foundation. 2021. "Improving artifact integrity across the software supply chain – SLSA." Available at (https://slsa.dev/)
SP80050	National Institute of Standards and Technology. 2021. "PRE-DRAFT Call for Comments: Building a Cybersecurity and Privacy Awareness and Training Program, SPS 800-50 Rev 1." Available at (https://csrc.nist.gov/publications/detail/sp/800-50/rev-1/draft)." Retrieved Sep. 25, 2021.
SP80052	National Institute of Standards and Technology. 2020. "Guidelines for the Selection, Configuration, and Use of Transport Layer Security (TLS) Implementations, SP 800-52 Rev. 2." Available at (https://csrc.nist.gov/publications/detail/sp/800-52/rev-2/final).
SP80053	National Institute of Standards and Technology. 2020. "Security and Privacy Controls for
SP80057	National Institute of Standards and Technology. 2020. "Recommendation for Key Management: Part 1 – General, SP 800-57 Part 1 Rev. 5." Available at (https://csrc.nist.gov/publications/detail/sp/800-57-part-1/rev-5/final)
SP800160	National Institute of Standards and Technology. 2018. "Systems Security Engineering." Available at (https://doi.org/10.6028/NIST.SP.800-160v1)
SP800161	"National Institute of Standards and Technology. 2021. ""Supply Chain Risk Management Practices for Federal Information Systems and Organizations."" Available at (https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-161.pdf)"
SP800172	"Enhanced Security Requirements for Protecting Controlled Unclassified Information: A Supplement to NIST Special Publication 800-171. Available at (https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-172.pdf)
SP800175B	National Institute of Standards and Technology. 2020. "Guideline for Using Cryptographic Standards in the Federal Government: Cryptographic Mechanisms. SP 800-175B Rev. 1." Available at (https://csrc.nist.gov/publications/detail/sp/800-175b/rev-1/final).

SP800181	National Institute of Standards and Technology, National Initiative for Cybersecurity Education. 2020. "Workforce Framework for Cybersecurity (NICE Framework)." Available at (https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-181r1.pdf)
SP800193	National Institute of Standards and Technology. 2018. "Platform Firmware Resiliency Guidelines, SP-800-193." Available at (https://csrc.nist.gov/publications/detail/sp/800-193/final).
SP800207	National Institute of Standards and Technology. 2020. "Zero-Trust Architecture, SP-800-207." https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-207.pdf
SSDF	National Institute of Standards and Technology. 2020. "Mitigating the Risk of Software Vulnerabilities by Adopting a Secure Software Development Framework (SSDF)." Available at (https://nvlpubs.nist.gov/nistpubs/CSWP/NIST.CSWP.04232020.pdf)
SWEBOK3	IEEE Computer Society. 2014. Guide to the Software Engineering Body of Knowledge. Available at (https://www.computer.org/education/bodies-of-knowledge/software-engineering/v3)
SYNOPSYS	Synopsys. 2021. "Synopsys Information Security Requirements for Vendors." Available at https://www.synopsys.com/company/legal/info-security.html
ZDNET	IBM, ZDNET. 2021. "Managing a Software as a Vendor Relationship: Best Practices". Available at (https://www.zdnet.com/article/managing-a-software-as-a-service-vendor-relationship-best-practices/)

3.5 Appendix E: Acronyms

Acronym	Meaning
ACQSEC	Acquisitions Security
AI	Artificial Intelligence
ATO	Authorization to Operate
CCB	Configuration Control Board
CIPAC	Critical Infrastructure Partnership Advisory Council
CISA	Cybersecurity & Infrastructure Security Agency
CNSSI	Committee on National Security Systems Instruction
CTI	Cyber Threat Intelligence
DUNS	Data Universal Numbering System
EO	Executive Order
EOL	End-of-Life
ESF	Enduring Security Framework
FARS/DFARS	Federal Acquisition Regulation/Defense Federal Acquisition Regulation
FOCI	Foreign Ownership, Control, Or Influence
IOC	Indicator of Compromises
IT	Information Technology
ITSM	IT Service Management
ML	Machine Language
NIST	National Institute of Standards and Technology (US DOC)
NSA	National Security Agency
NTIA	National Telecommunications and Information Administration (US DOC)
ODNI	Office, Director National Intelligence
PO	Prepare Organization
PS	Protect Software
PW	Produce Well-Secured Software
RFI	Request for Information
RFP	Request for Proposal
RV	Respond to Vulnerabilities
SBOM	Software Bill of Material
SCRM	Supply Chain Risk Management
SDLC	Software Development Lifecycle
SLSA	Supply-chain Levels for Software Artifacts
SOC	Security Operations Center

SOW	Statement of Work
SSDF	Secure Software Development Framework
USG	United States Government