sysdig SECURE EVERY SECOND.

E-BOOK

Supply Chain Security Best Practices

Table of Contents

CHAPTER 01

About This Book

Welcome to Supply Chain Security Best Practices! This ebook is your comprehensive go-to resource for exploring the fundamentals of software supply chain security. Whether you are a newcomer seeking to understand the fundamentals of Software Supply Chain or a seasoned cyber security professional looking for insights to secure your organization against supply chain attacks, this ebook offers invaluable tips to help elevate your game.

Inside these pages, you will find useful tips to help you navigate the complexities of Supply Chain Security and best practices to secure it. Sysdig's mission is to empower you with the knowledge and tools you need to secure against supply chain attacks.

Unveil the different tactics, techniques and procedures cyber criminals use to exploit the whole chain with our comprehensive ebook. Learn from real-life scenarios to fortify every stage of your development cycle and increase your detection efficacy.

In this eBook, you'll learn:

- The basics of supply chain security.
- How to secure your suppliers and vendors.
- What are the main supply chain security threats and how can you protect from them?
- How industry and governments attempt to organize a knowledge base.
- Best practices for ensuring supply chain continuity.

Thank you for choosing the Supply Chain Security Best Practices ebook. We're excited to join you on your quest to defend against supply chain attacks.

CHAPTER 02

Software Supply Chain

The software supply chain is similar to other activities or industries. Some resources are consumed, then transformed, through a series of steps and processes, and finally supplied as a product or service.

In software, the raw materials are common libraries, code, hardware, and tools that are commonly deployed as either a user-facing application, a service (starting over with the same supply chain loop), or a dependent package.

Let's look at the example below.

To produce the applications (illustrated), we need to compile a source code and consume information from third-party services.

The source code may depend on external libraries, which are generated from another codeset, with other similar dependencies, and the chain goes on.

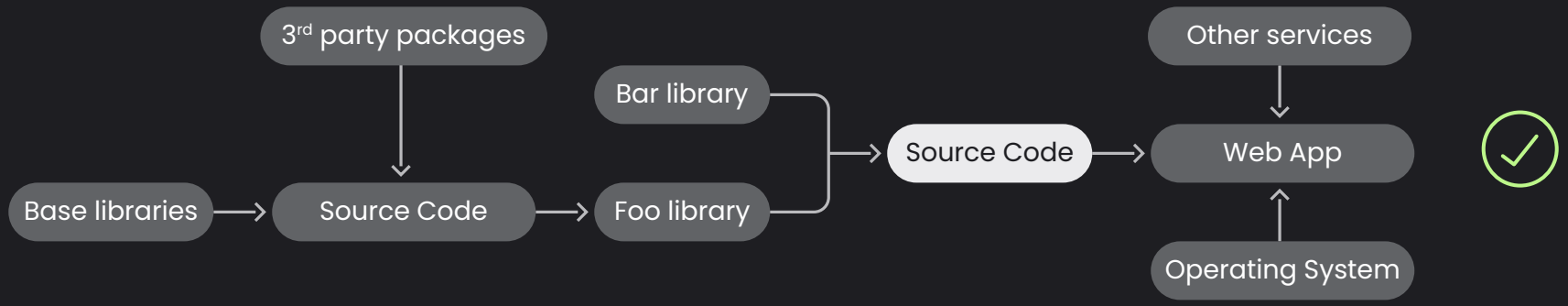


Figure 1

A typical software supply chain

Security Aspects

Software supply chain security risks are summarized below:

- **Tampering** or manipulation, a malicious element is inserted into the chain by a threat actor.
- **Vulnerabilities**, vulnerable code that cause an implicit risk in libraries or packages. These vulnerabilities are often unintentional and can remain dormant for a long time.

Chain Implications

A successful attack in any segment of the supply chain link propagates the compromised code downstream, completely unnoticed. In fact, cybercriminals focus on a software vendor compromise by injecting malware or vulnerability at an intermediate step to exploit the consumers.

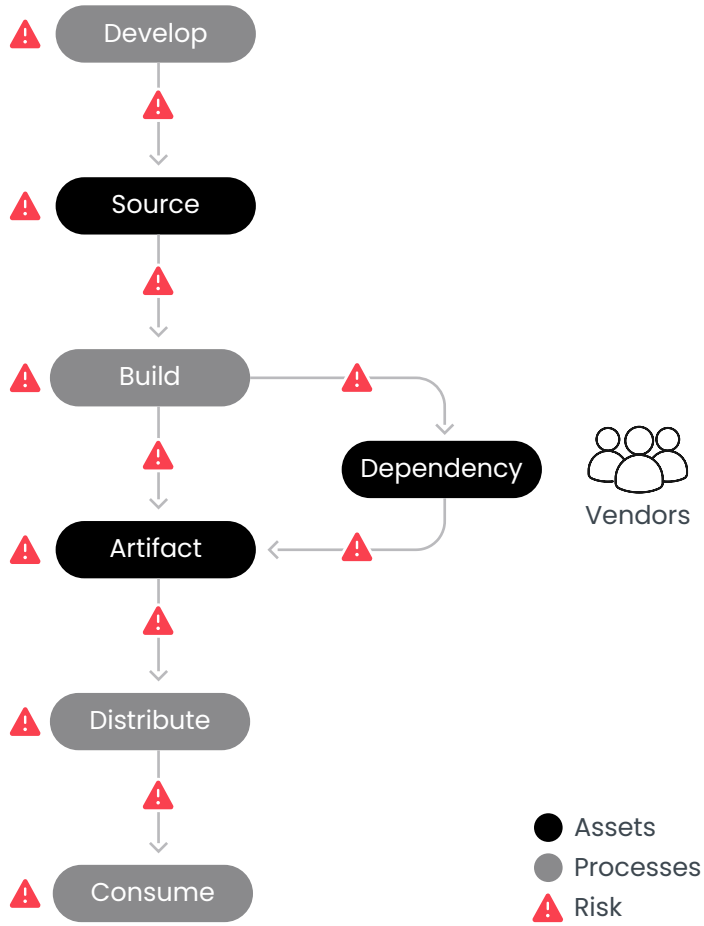
A few security measures can be applied:

1. **Inputs:** Assess the origin and integrity of libraries and package dependencies, third-party tools, software, services, or any artifact you are consuming, either public or from a private vendor.
2. **Outputs:** Guarantee the integrity and traceability of your deliverables and provide ways to verify the components downstream.
3. **Processes and infrastructure:** Protect your network, identities, credentials, signature keys, repositories, and processes.

Next, we'll review a few common supply chain attacks, with recent examples, and learn the tips and best practices to mitigate risks.

Figure 2

Each link in Supply Chain is vulnerable



CHAPTER 03

What a Supply Chain Attack Looks Like

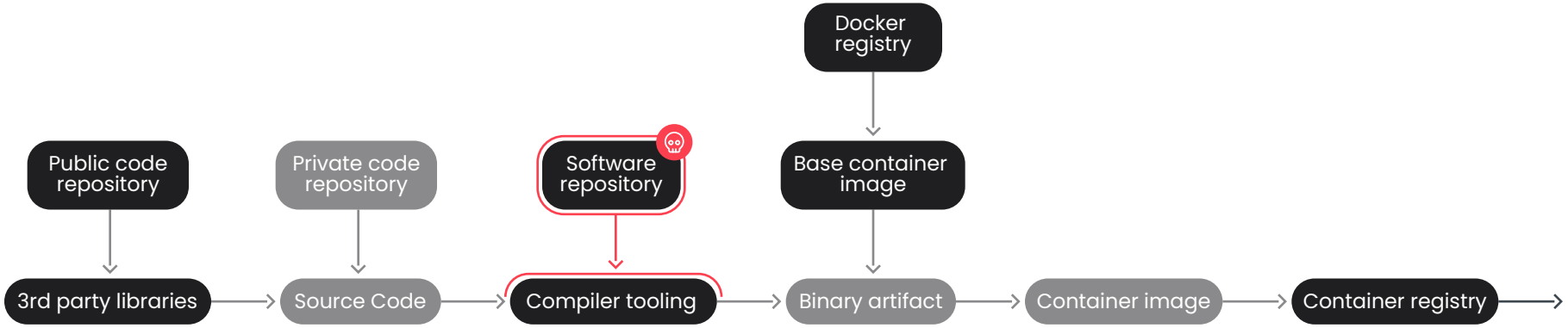
Figure 3

A compromised
Software Suply Chain

Let's dive deeper and focus on the source code.

Assume your source code, hosted in a private git repository, is part of your infrastructure or a SaaS provided by a vendor, along with compiler tools, base container image registries, etc.

Some dependencies hosted within public repositories, like Docker Hub or Quay.io, could be compromised. Our application will also be published here as a container image.



The source and application codes, binary artifacts, and the container images (highlighted in grey) exist in your private git repository. However, several other components (highlighted in black) are public services and resources, or provided by other vendor companies.

A software supply chain attack can either target you directly, or any upstream elements (like external dependencies or provided services). The consequences of exploiting any of these compromised elements are unpredictable and potentially disastrous for enterprises.

Real-life cases

Trends show that supply chain attacks are increasing at an exponential rate – the number of malicious packages have tripled since 2022.

Fortunately, not every attack has a big enough impact to appear in the newspaper, but let's analyze some of the most relevant and recent ones. Many other examples of different types of supply chain attacks are also collected by the CNCF in its Catalog of Supply Chain Compromises.

Credential stealer (mathjs-min)

Cybercriminals modified the widely-used Javascript library mathjs (a Javascript math library) and uploaded it to NPM (Node Package Manager) as a "minified" version called mathjs-min, which contained a Discord token grabber.

To make the malicious package appear legitimate, the author copied the README directly from the genuine mathjs package. Strangely, the author also included a link to its forked GitHub repository, which reveals its intentions through its commit history.

This incident serves as a stark reminder of the potential security risks that come with using open source software, especially as threat actors are evolving their tactics.

Fake commits (Dependabot)

Scanners detected non-typical commits carrying malicious code to hundreds of Github repositories, contributed by Dependabot – a bot maintained by GitHub that automatically submits pull requests to update project dependencies.

Threat actors fabricated those Dependabot commit messages to appear as an automated contribution in the commit history and the repository maintainers subsequently merged it. The malicious code exfiltrates defined secrets to a C2 server and threat actors modify any existing javascript files in the attacked project with a web-form password-stealer malware code effecting any end-user submitting its password in a web form. This incident teaches us to be diligent while downloading the code, even from trusted places like GitHub.

Account takeover (Packagist.org)

Cybercrimnials accessed four user accounts that had been inactive on Packagist.org for a period of time but still had access to a total of 14 packages. Per reports, no malicious code was distributed, and the user accounts were detected and reverted. This incident teaches us to never reuse passwords and enforce multi-factor authentication wherever possible.

CHAPTER 04

Securing the Software Supply Chain

Attacks to the software supply chain are broadening the attack surface of companies, as their security does not only depend on internal procedures. Now, they also need to ensure that third parties (including software, hardware, services, etc.) are not a gateway to attackers who can affect them.

Like traditional security, it is impossible to secure everything, especially as new kinds of software supply chain attacks are being discovered continuously. But let's explain where you need to focus on each of the layers to be as protected as possible.

Manage inputs of your software development

As a consumer of software artifacts, hardware, and services from multiple providers, ensure that your providers apply a high level of security.

The software supply chain is becoming so critical that in May 2021, the president of the United States issued an Executive Order targeted to improve the nation's cybersecurity. As a result of the order, the NIST produced the Recommended Minimum Standards for Vendor or Developer Verification (Testing) of Software Under Executive Order (EO) 14028 guide, which recommends minimum standards for verification by software vendors or developers.

Because it is impossible to achieve and guarantee 100 percent security, please also pay attention to any risks or incidents impacting your providers, and be ready to take quick corrective action in case a compromised component is either detected or reported.

Secure software development in your company

With regard to code and development processes, the aforementioned Recommended Minimum Standards for Vendor or Developer Verification (Testing) of Software Under Executive Order (EO) 14028 includes the following set of techniques as the minimal safety requirements for the software development life cycle. Make sure you don't miss any of them:

- Apply threat modeling to identify key or potentially overlooked testing targets.
- Automated testing.
- Code-based (static) analysis, using a code-scanner, and review for hard-coded secrets.
- Dynamic analysis, with built-in checks and protections, black-box and fuzzy testing, web-app scanner, etc.
- Adopt vulnerability prioritization strategies combining Runtime Insights with other contexts, like exploitability, to identify what incurs high risk in your environment. CVSS scores and vulnerabilities severity don't take into account environment specifics that impact the actual risk exposure.
- Apply similar checks to included software (third-party dependencies).
- Fix critical bugs as soon as possible.

With Infrastructure as Code (IaC), threats in the software supply chain now potentially target not just software and applications, but also the underlying infrastructure. The same software verification principles should be applied to the deployment and management of infrastructure as code, shifting security left.

But security must be pervasive in all development stages, as well as the whole company culture and practices. Securing the code and the development process is clearly not enough, as outlined by multiple organization researches and guides:

- The Defending Against Software Supply Chain Attacks guide from Cybersecurity and Infrastructure Security Agency considers that the Software Supply Chain Lifecycle has six phases where "software is at risk of malicious or inadvertent introduction of vulnerabilities":
  - Design
  - Development and production
  - Distribution
  - Acquisition and deployment
  - Maintenance
  - Disposal
- The Cloud Native Computing Foundation (CNCF) keeps a living, community-maintained document: the Software Supply Chain Security Paper. This paper aims to contribute to the community with a series of recommended practices, tooling options, and design considerations that can reduce the likelihood and overall impact of a successful supply chain attack.
- The CNCF, Linux Foundation, VMware, Intel, Google, and others are also working on SLSA – Supply-chain Levels for Software Artifacts, a security framework, and a common language for increasing levels of software security and supply chain integrity for anyone working with the software. Each level provides an increasing degree of confidence, a way to say that software hasn't been tampered with and can be securely traced back to its source.

Kubernetes and containers are so common nowadays that NSA/CISA also released a Kubernetes Hardening Guidance, highlighting "Supply chain risks" as one of three sources of compromises, and proposing the following hardening measures and mitigations:

- Scan containers and Pods for vulnerabilities or misconfigurations.
- Run containers and Pods with the least privileges possible.
- Use network separation to control the amount of damage a compromise can cause.
- Use firewalls to limit unneeded network connectivity and encryption to protect confidentiality.
- Use strong authentication and authorization to limit user and administrator access, as well as to limit the attack surface.
- Use log auditing so that administrators can monitor activity and be alerted to potential malicious activity.
- Periodically review all Kubernetes settings and use vulnerability scans to help ensure risks are appropriately accounted for and security patches are applied.

Safe distribution of your software

Everything you apply when choosing your best providers with a high level of confidence will apply to your company when acting as a supplier for other companies or for end-users. Even if you correctly implement security into every step and process of your software supply chain life cycle, you still need to manifest it and add specific measures to the delivered artifacts.

- Provide evidence of regulatory compliance and security certifications within your organization that applies to the software supply chain.
- Add a Software Bill of Materials (SBOM) as a way to track dependencies and third-party sources of compromise in your software.
- Include digital signatures to prevent tampering and verify the source of your artifacts.
- Use safe distribution channels, encrypted communications, and trusted hosting or storage infrastructure.

Why Shifting Left and SBOM are not enough

While the shifting-left methodology and SBOM can certainly help to improve the security of software supply chains, they may not be enough on their own to fully secure a software supply chain.

Shift-left methodology

The shifting-left methodology refers to the practice of integrating security and risk management activities earlier in the software development lifecycle, rather than waiting until later stages of development or after the software has been deployed. By identifying and addressing security risks earlier in the development process, organizations can reduce the likelihood of security incidents and minimize the impact of any incidents that do occur. However, the shifting-left methodology alone does not necessarily address all of the risks associated with software supply chains, such as third-party dependencies, software updates, and code changes made after deployment.

Software Bill of Materials (SBOM)

SBOMs are an important component of software supply chain security, as they provide a comprehensive list of all of the components and dependencies in a software system. This can help organizations identify and manage risks associated with software supply chains, such as vulnerabilities in third-party components or outdated software versions. However, SBOMs are not a panacea for software supply chain security. For example, they may not capture all of the components in a software system, and they do not necessarily provide information on how the components were developed or how they interact with other components in the system.

What is the ideal approach for securing a supply chain?

The ideal approach should also provide a complete view of risk and ways to prioritize it (i.e., exploitability, exposure, runtime insights).

To fully secure a software supply chain, organizations need to take a comprehensive approach that addresses all of the risks. This may include implementing secure development practices, conducting regular security testing, implementing technical controls to secure the software supply chain, and establishing policies and procedures for managing software supply chain risk. Additionally, organizations should work with their vendors and partners to ensure that they are also implementing secure software development and supply chain management practices, as vulnerabilities in third-party components can also pose significant risks to the security of a software supply chain.

CHAPTER 05

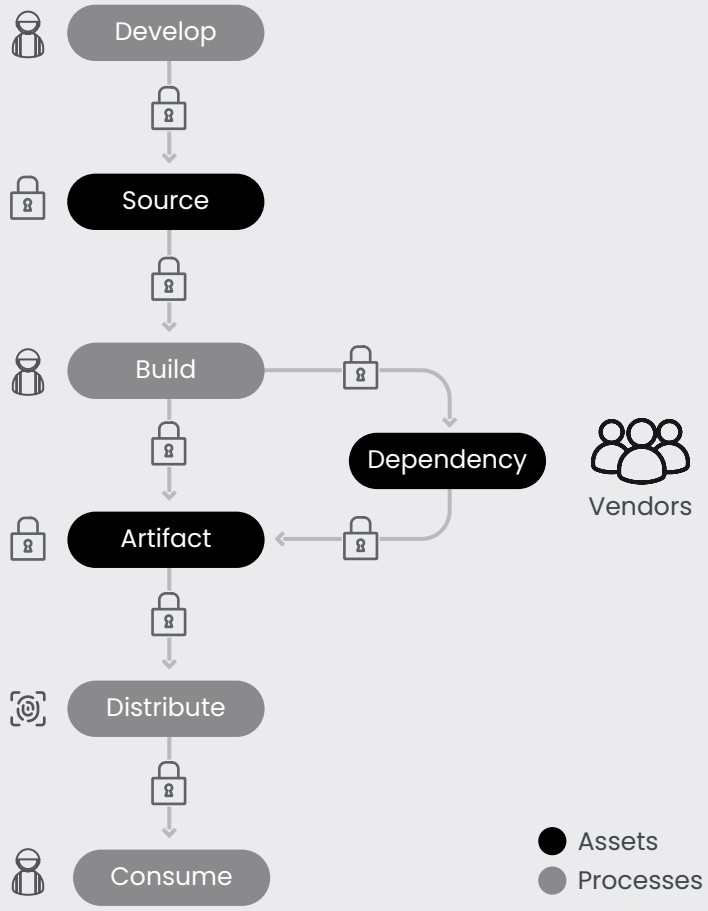
Best Practices for Securing the Supply Chain

It's important to have a plan of action for addressing risks. The ideal approach is to embrace frameworks like SLSA. However, we have compiled a list of best practices for securing your supply chain that anyone can use as a reference.

1. **Foundational security** is critical:
   Follow zero trust and proactively manage the security posture of your supply chain ecosystem. Keep the whole organization trained and engaged with security as a priority. This point is always undervalued.
2. **Establish a Secure Development Lifecycle (SDL):** A set of processes that ensure security is built into the software development process. This includes security testing, code review, and vulnerability scanning. Implementing a SDL helps to identify and mitigate vulnerabilities.
3. **Apply risk prioritization:** Prioritization mechanisms like runtime insights are key to reducing the Time To Resolution (TTR).
4. **Verify the integrity of software and firmware:** Use digital signatures to verify the integrity and provenance of software and firmware. This helps to ensure that the code has not been tampered during the distribution process. Implant policy-based verifications across the pipeline as well as in the Kubernetes Admission controller.
5. **Use multi-factor authentication (MFA):** MFA adds an extra layer of security to user authentication. Use multi-factor authentication for all users who have access to critical systems.
6. **Use encryption:** Use encryption to protect sensitive data in transit and at rest. This includes encrypting data in databases, encrypting network traffic, and encrypting data on storage devices.
7. **Secrets management strategy:** Vault tools are good allies to store and rotate keys and enable secured automations.

Figure 4

Secure each link of your Supply Chain



8. **Implement access controls:** Implement access controls to restrict access to sensitive data and systems. Use role-based access control (RBAC) to ensure that users only have access to the data and systems they need to perform their job functions.
9. **Continuous security assessments:** Conduct regular security assessments to identify vulnerabilities and risks in the supply chain. This includes vulnerability scanning, penetration testing, and risk assessments.
10. **Maintain an up-to-date inventory:** Maintain an up-to-date inventory of all hardware and software assets in the supply chain. This helps to ensure that all systems are accounted for and that security updates are applied in a timely manner.
11. **Establish incident response procedures:** Establish incident response procedures to ensure that security emergencies are detected and responded to in a timely manner. This includes runbooks, procedures for reporting security incidents, containing security incidents, and recovering from security incidents.
12. **Monitor for security events:** Use security event monitoring tools to detect security events in the supply chain. This includes monitoring network traffic, system logs, and user activity.
13. **Maintain vendor relationships:** Maintain strong relationships with vendors in the supply chain. This includes requiring vendors to adhere to security best practices and conducting regular security audits of vendor systems.

Implementing these best practices can help to secure the supply chain and mitigate the risk of security incidents.

CHAPTER 06

Supply Chain Regulations and work groups

The future of software supply chain regulations is likely to involve a greater focus on supply chain security and risk management. The increasing reliance on software in all areas of business and society means that the security of software supply chains is becoming a critical issue.

There is growing awareness of the risks associated with insecure software supply chains, and governments and industry organizations are taking steps to address these risks. In the U.S., there are several initiatives underway to improve the security of software supply chains.

Cybersecurity and Infrastructure Security Agency (CISA)

In the U.S., CISA has launched the Software Supply Chain Integrity initiative, which aims to promote best practices and guidelines for securing software supply chains.

The initiative has three main goals:

1. **Improving software supply chain security:** CISA is working to develop best practices and guidelines for securing software supply chains. This includes developing guidance for organizations to assess and manage software supply chain risk, as well as recommendations for software vendors to ensure the security of their products.
2. **Promoting awareness and education:** CISA is working to raise awareness of the risks associated with insecure software supply chains and to educate organizations on how to improve their supply chain security. This includes providing training and resources to help organizations implement best practices and improve their software supply chain risk management.
3. **Engaging stakeholders:** CISA is engaging with stakeholders across all sectors of the economy to promote best practices and encourage collaboration on software supply chain security. This includes working with industry groups, government agencies, and other organizations to develop and implement best practices and guidelines for securing software supply chains.

Overall, the Software Supply Chain Integrity initiative is an important step in improving the security of software supply chains in the U.S. By promoting best practices and engaging stakeholders, CISA is helping to ensure that organizations across all sectors of the economy are better equipped to manage software supply chain risk and secure their software supply chains.

The National Institute of Standards and Technology (NIST)

Similarly, NIST has developed guidelines and standards for software supply chain risk management to help organizations better manage and mitigate the risks associated with insecure software supply chains. NIST's guidelines and standards provide a framework for managing software supply chain risks, including:

1. **Identifying and assessing software supply chain risks:** NIST provides guidance on how to identify and assess risks associated with software supply chains, including the risks associated with software development, distribution, and maintenance.
2. **Implementing risk management strategies:** NIST provides guidance on how to implement risk management strategies to mitigate software supply chain risks. This includes guidance on developing policies and procedures for managing software supply chain risks, as well as recommendations for technical controls that can be implemented to secure software supply chains.
3. **Monitoring and responding to software supply chain risks:** NIST provides guidance on how to monitor software supply chain risks and respond to security incidents. This includes guidance on incident response planning, as well as recommendations for implementing continuous monitoring programs to detect and respond to software supply chain security incidents.

NIST also defined SSDF (Secure Software Development Framework), a set of fundamental software development practices to help producers reduce the number of vulnerabilities.

In Europe, there has been a similar focus on improving software supply chain security through stricter regulation on the software supply chain vendors.

European Union Agency for Cybersecurity (ENISA)

In the EU, the agency ENISA had published guidelines for software supply chain security to help organizations better manage the risks associated with insecure software supply chains. ENISA's guidelines provide recommendations for assessing and managing software supply chain risks, as well as guidance on implementing technical controls to secure software supply chains.

Additionally, the European Commission is working on a proposal for a new cybersecurity certification framework that includes requirements for software supply chain security. The proposed framework aims to establish a common approach to cybersecurity certification across the EU and to improve the overall security of digital products and services. The framework includes specific requirements for software supply chain security, such as requirements for secure software development practices and supply chain risk management.

There is also growing pressure on software vendors to take responsibility for the security of their solutions. Customers are increasingly demanding that vendors provide assurance that their software has been developed and delivered securely. This includes implementing secure software development practices, conducting regular security testing, and providing transparency into their software supply chains.

OpenSSF, SLSA, and S2C2F

Some industry groups, such as the Open Source Security Foundation (OpenSSF), are working to promote best practices for secure software development and supply chain management. These efforts aim to improve the overall security of software supply chains and to ensure that vendors take responsibility for the security of their solutions.

The OpenSSF Supply Chain working group promotes and develops some frameworks:

- SLSA (Supply Chain Levels for Software Artifacts) is a security framework oriented to software producers. It is similar to a checklist with controls to grant integrity, and secure packages and infrastructure. SLSA provides a set of incremental levels of guarantee.

- S2C2F (Secure Supply Chain Consumption Framework) is a security framework oriented to threats (tactics and techniques) used by adversaries to compromise OSS packages. It is also organized into levels.

  Both frameworks are complementary.

- FRSCA (Factory for Repeatable Secure Creation of Artifacts) provides a suite of pipelines and pipeline abstractions to help secure the supply chain by securing build pipelines.

In summary, the future of software supply chain regulations is likely to involve greater attention to supply chain security and risk management. There are already initiatives and guidelines in place in the U.S. and Europe, and there is growing pressure on software vendors to take responsibility for the security of their solutions.

CHAPTER 07

Conclusion

In this Supply Chain eBook, we have presented:

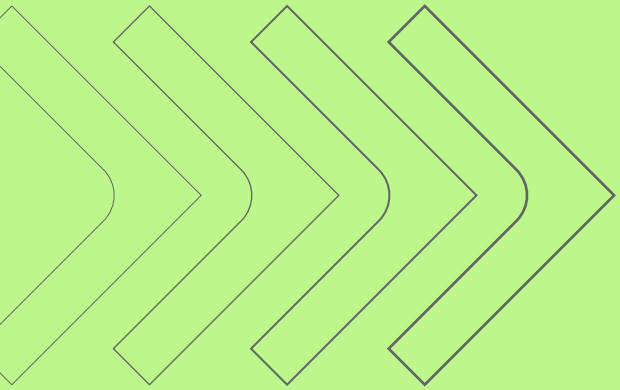
- The basics of supply chain security and why it is so important.
- Mentioned how to secure suppliers and vendors.
- Highlighted the importance of risk prioritization through the entire SDLC and that supply chain security is more than shift-left and SBOMs.
- Some information about current landscape, frameworks, and work groups.
- Provided best practices for ensuring supply chain continuity.
- Gave insights into how supply chain regulations are evolving.

We hope that you found this information useful as you navigate the complexities of securing your supply chain. It can be a challenging journey, but with the best practices and tips presented in this ebook, you can take steps to protect your supply chain from cyber threats and other risks.

To further assist you in securing your supply chain, we recommend exploring the resources available on our Learn Cloud Native webpage. You can also take advantage of our expertise by using our Sysdig Platform, which converges security and compliance with performance and capacity monitoring to create a secure DevOps workflow. Our platform uses the same data to monitor and secure, allowing you to identify where a problem occurred and why it happened. You can use this information to investigate and troubleshoot performance and security issues.

If you are interested in learning more about securing your supply chain, we encourage you to check out the official Sysdig blog, where our experts are always posting new tips and recommendations.

With both of these resources in hand, you will be well on your way to understanding how to secure and maintain the integrity of your supply chain!



See how Sysdig helps you secure every second.

Take the next step.

REQUEST A DEMO →

sysdig

E-BOOK

EBK-008 REV. B 3/24

About Sysdig

In the cloud, every second counts. Attacks move at warp speed, and security teams must protect the business without slowing it down. Sysdig stops cloud attacks in real time, instantly detecting changes in risk with runtime insights and open source Falco. Sysdig correlates signals across cloud workloads, identities, and services to uncover hidden attack paths and prioritize real risk. From prevention to defense, Sysdig helps enterprises focus on what matters: innovation.

Sysdig. Secure Every Second.