harness

Definitive Guide to

SECURE SOFTWARE DELIVERY

Contents

harness

Contents

Introduction

If you're reading this ebook, you're well aware of how much more decentralized and complex software development has become over the last decade or two. You're also aware that the speed in which organizations build and deploy modern applications exposes them and their users to a wide range of security and compliance risks. As a result, software-producing organizations grapple with two challenging phenomena:

- **Traditional application security practices are not effective in a modern DevOps world.** When security scans are run only at the end of the software delivery lifecycle (either right before or after a service is deployed), the ensuing process of compiling and fixing vulnerabilities creates massive overhead for developers. Overhead that degrades velocity and puts production deadlines at risk.
- **Regulatory pressure to ensure the integrity of all software components is ramping up dramatically.** Applications are built with an increasing number of open source software (OSS) components and other 3rd party artifacts, each of which can introduce new vulnerabilities into the application. Attackers seek to exploit these components' vulnerabilities, which also puts the software's consumers at risk.

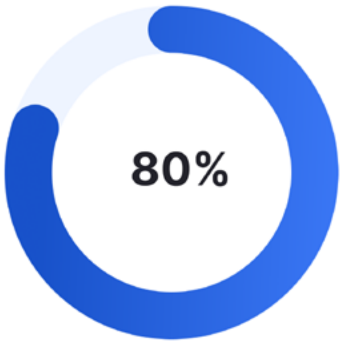
Software represents the largest under addressed attack surface that organizations face. The current threat environment, coupled with the drive to deliver applications faster, compels organizations to integrate security throughout the software development lifecycle in ways that don't degrade developer productivity. This practice is formally known as DevSecOps.

Delivering secure software– the outcome of an effective DevSecOps program– is a huge undertaking. It requires significant cultural changes across multiple functions to drive shared responsibility, collaboration, transparency, and effective communication. It also requires the right set of tools, technologies, and use of automation and AI to secure applications at the speed of development. Implemented correctly, DevSecOps becomes a major success factor in delivering secure software.

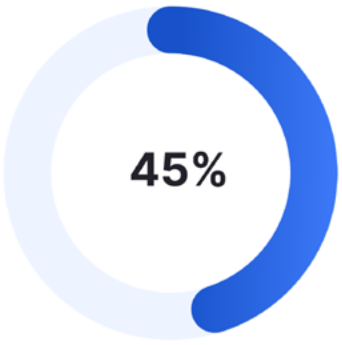
We'll provide an overview of what's required from a tools, technologies, and process perspective to deliver software that is more secure, faster.

By the end, you should feel comfortable and confident in moving your organization towards adopting all of the best practices designed to strengthen your security posture and minimize risk for your business.

Key Statistics

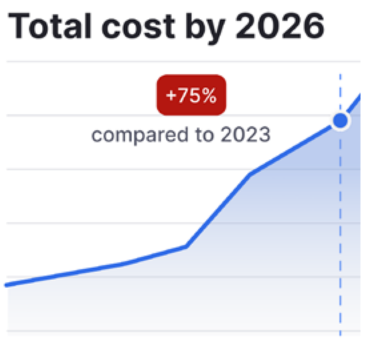


More than 80% of software vulnerabilities are introduced through open source software (OSS) and 3rd party components



Digital supply chain attacks are becoming more aggressive, sophisticated, and diverse. By 2025, 45% of organizations will have experienced at least one

GARTNER RESEARCH



Total cost of software supply chain cyber attacks to businesses will exceed $80.6 billion globally by 2026, up from $45.8 billion in 2023

JUNIPER RESEARCH

What is DevSecOps and Why is it Essential for Secure Software Delivery?

What is DevSecOps and Why is it Essential for Secure Software Delivery?

DevSecOps, short for development, security, and operations, is an approach to software development that integrates security practices throughout the entire software development lifecycle. It emphasizes collaboration and communication between development teams, security teams, and operations teams to ensure that security is built into every stage of the software development process.

Within the context of software development pipelines, DevSecOps aims to “shift security left”, which essentially means as early as possible in the development process. Quite frankly, it involves integrating security practices and tools into the development pipeline from the very beginning. By doing so, security becomes an integral part of the software development process rather than a late-stage add-on.

This approach makes it significantly easier for organizations to identify and resolve security vulnerabilities early on, and meet regulatory obligations. It’s also important to note that DevSecOps is built upon a culture of collaboration and shared responsibility. It breaks down silos and encourages cross-functional teams to work together towards a common goal of building more secure applications at high velocity.

Who is Responsible for Secure Software Delivery?

Who is Responsible for Secure Software Delivery?

In DevSecOps programs, there are several primary stakeholders. These stakeholders collaborate consistently to achieve the common goal of building secure and reliable software. Let's take a look at who's involved and what the scope of their roles are below:

- **Development Teams:** Dev teams are responsible for writing code, designing software architecture, and implementing new features. In DevSecOps, dev teams need to embrace security as an integral part of their work. They should understand and follow secure coding practices, conduct code reviews, and integrate security testing into their processes. By incorporating security from the beginning, dev teams can minimize vulnerabilities and ensure that the software is built with security in mind.
- **Security Teams:** Application security teams, more commonly referred to as AppSec, are responsible for identifying and mitigating software and infrastructure security risks and ensuring compliance with industry standards and regulations. In DevSecOps, security teams collaborate closely with developers to provide guidance on secure coding practices, perform security assessments, and conduct penetration testing. They also help in defining security policies, monitoring environments and systems for potential threats, and responding to security incidents.
- **Operations Teams:** Operations teams at large are responsible for deploying, managing, and maintaining the software in production environments. With traditional DevOps, essentially IT operations support developers with pipelines and tooling, along with the underlying infrastructure resources. In DevSecOps, Ops works closely with development and security teams to ensure that the software is deployed securely and that proper security controls are in place. They collaborate on infrastructure design, configuration management, and continuous monitoring of systems for any security vulnerabilities or anomalies. Operations teams also play a critical role in incident response and recovery in case of security breaches or system failures.

- **Quality Assurance Teams:** Quality assurance teams, better known as QA, are responsible for testing and ensuring the quality of the software. In DevSecOps, QA teams collaborate with development and security teams to incorporate security testing into their test plans. They perform functional testing, performance testing, and security testing to identify any vulnerabilities or weaknesses in the software. By including security testing as part of their QA processes, they help in identifying and addressing security.
- **Management and Leadership:** Management and leadership teams have a vital role in driving the adoption of DevSecOps practices within the organization. They set the vision, allocate resources, and establish a culture of security and collaboration. Management teams need to prioritize security initiatives, provide training and education on security best practices, and ensure that security is integrated into the organization's overall strategy. They also play a crucial role in fostering communication and collaboration between different teams and departments.

As you can see, the list of stakeholders is quite long. Though it's ideal for collaboration, getting all of these teams together is not easy. Each stakeholder group has their own set of agendas and direction to comply with, and sometimes these goals conflict with those of the teams they need to work with. This is just one of a few notable challenges that organizations experience when building and maintaining a meaningful DevSecOps process. We'll cover these issues in more detail in the next section.

Secure Software Delivery Challenges

Secure Software Delivery Challenges Challenges

As we alluded to in the previous section, there are several challenges that organizations face when implementing a DevSecOps practice. These challenges can hinder the successful integration of security controls into the software development process. Understanding these challenges is crucial for organizations to address them effectively and ensure the smooth adoption of DevSecOps principles.

Siloed Work Culture

One of the main obstacles in achieving a working DevSecOps model is organizational culture. Traditionally, development, security, and operations teams have worked in silos, with limited communication and impromptu collaboration. To overcome this challenge, organizations need to foster a culture of shared responsibility and collaboration. This involves breaking down barriers between teams, promoting open communication, and encouraging a mindset where security is everyone's responsibility. By creating a culture that prioritizes security and encourages collaboration, organizations can effectively address the challenges associated with DevSecOps implementation.

Security Tool Management Complexity

Building secure software from raw code to development to deployment becomes complex because of the myriad of available tools and security scanners. Organizations tend to use multiple SAST, DAST, secrets, and container scanning tools in their development environments. The outputs from all these tools are disparate and there is no uniform data format or language.

Consequently, developers don't get a deduplicated and prioritized list of vulnerabilities to remedy, which software teams must normalize all the output, track exemptions, and verifying fixes. This all requires synchronization between DevOps and security teams; it also takes a lot of time and labor away from other work. In addition, DevOps and security teams need to act on the information provided from security testing, but ensuring that these are the only vulnerabilities is challenging.

Organizations need to choose tools that align with their specific requirements and integrate well with the underlying devops platform and pipelines. Automation is a key aspect of DevSecOps, and organizations should seek a platform approach where security tooling and workflows are integrated seamlessly within the devops platform itself, as opposed to applying a collection of point solutions that are complex to manage.

Delayed Feedback of Security Information

In many DevOps programs, most security testing is done right before code has reached production. All of the release stages where security testing could have been applied are past, and fixing security issues requires reworking each stage. This results in a delayed feedback loop of vulnerability information from security teams back to developers. Developers need to move forward but by the time the security testing feedback arrives it could be days or weeks later, forcing developers to drop their current projects to hunt for and fix vulnerable code.

Lack of Open Source Software Visibility

Modern applications comprise an increasing number of open source software components and 3rd party artifacts, each of which comprise vulnerabilities that pose security risks to the application's users. Lack of visibility into the potential vulnerabilities of these components and the inability to verify their integrity has led to several high profile catastrophic breaches in recent years. In 2021, Log4j, an open-source logging framework maintained by Apache and used in a myriad of different applications, was the root of exploits that put thousands of systems at risk. Log4j's communication functionality was vulnerable and thus provided an opening for an attacker to inject malicious code into the logs which could then be executed on the system. After its discovery, security researchers saw millions of attempted exploits, many of which turned into successful denial-of-service (DoS) attacks.

Guiding Principles for Delivering Secure Software

Guiding Principles for Delivering Secure Software

At a high level, building and running an effective DevSecOps program means that your organization is able to operate a secure delivery platform, test for software vulnerabilities, prioritize and remediate vulnerabilities, prevent the release of insecure code, and ensure the integrity of software and all of its artifacts. Below are detailed descriptions of the elements and required capabilities to achieve a successful DevSecOps practice.

Establish a Collaborative Culture That Makes Security a Shared Responsibility

The success of any DevSecOps practice is really in the hands of its stakeholders, so before setting out to acquire, configure and deploy new tools and technologies,

If your organization builds, sells, or consumes software (which today is every conceivable organization on the planet), then every single employee has an impact on the overall security posture– not just those with 'security' in their titles. At its core, DevSecOps is a culture of shared responsibility, and operating with a common security-oriented mindset determines how well DevSecOps processes fit into place and can drive better decision making when choosing DevOps platforms, tooling, and individual security solutions.

Mindsets don't change overnight, but alignment and a sense of security accountability can be achieved through the following:

- **Commitment to regular internal security training– tailored to DevSecOps– that includes developers, DevOps engineers, and security engineers. Skills gaps and needs shouldn't be underestimated.**
- **Developer adoption of secure coding methodologies and resources**
- **Security engineering contributes to application and environment architecture, design reviews. It's always easier to identify and fix security issues early in the software development lifecycle.**

Break Down Functional Silos and Collaborate Continuously

Since DevSecOps is a result of the confluence of software development, IT operations, and security, breaking down silos and actively collaborating on a continuous basis is critical for success. Typically, DevOps-centric organizations operating without any formal DevSecOps framework see security entering the picture like an unwelcome party crasher. Process changes or tooling that is suddenly imposed (as opposed to collaboratively chosen and instantiated) invariably results in development pipeline friction and unnecessary toil for developers. A common scenario involves security mandating additional application security checks without consideration for their placement within the pipeline, or for how much workload is required to process scanner output and remediate vulnerabilities, which inevitably falls to developers.

- **Driving collaboration and operating as a cohesive DevSecOps team involves:**
- **Defining and agreeing upon a set of measurable security objectives, such as mean time to remediation and % reduction in CVE alert noise.**
- **Involvement from software developers and DevOps teams throughout the evaluation and procurement processes for new security tools**
- **Ensuring no DevSecOps process has a single functional gatekeeper**
- **Iteratively optimizing tooling choices and security practices for developer productivity and velocity**

Build and Deploy More Secure Code

Adopt and Enforce Secure Coding Methodologies

Strengthening application security posture starts with writing robust and secure code. Development teams should endeavor to adopt secure coding guidelines and best practices to write code that is more resilient against common security threats. This includes practices such as input validation, output encoding, and proper handling of sensitive data. By following secure coding practices, developers can minimize the introduction of vulnerabilities into the codebase.

Shift Security Left

Implementing shift-left security is a crucial step in securing application code as it moves through development pipelines. This approach involves integrating security practices early in the software development lifecycle, starting from the initial stages of coding and extending throughout the entire development and deployment process. By shifting security testing further left, organizations can identify and address vulnerabilities at an early stage, reducing the risk of security breaches and ensuring the delivery of secure applications.

Shifting security left successfully starts with the integration and orchestration of different types of security scanners throughout development pipelines. There are several categories of application security tests that DevSecOps teams need to adopt and employ in order to catch and remediate vulnerabilities throughout the software development lifecycle. The techniques employed by each type of security scanner are complimentary. Combined, they are very effective in surfacing known security issues before an application hits production.

Application Security Testing (AST)

SAST (Static Application Security Testing) Scanners

Static application security testing (SAST) involves analyzing application source code to detect security vulnerabilities that could potentially be exploited. SAST is applied early in the SLDC, prior to code being compiled. SAST scanners should be run on code on a regular basis, such as during periodic builds, at each code check-in, or during a code release. Catching and fixing vulnerabilities in the code base at an early stage has a dramatic impact on the quality and security posture of the final application.

SCA (Software Composition Analysis)

SCA tools are used to identify open source software within a code base, for the purpose of evaluating security, license compliance and overall code quality. As the vast majority of modern applications are built with open source software components, it is very important to be aware not only of the inherent security risks associated with a particular artifact, but also of licensing considerations regarding the use of that artifact or library.

Container Scanning

A rapidly-growing number of modern applications are built as collections of small composable elements called containers. A container packages up a short piece of code– the container image– along with all of its dependencies, binaries, and libraries. Container scanning tools are purpose built to analyze containers and their contents for known security issues.

DAST (Dynamic Application Security Testing) Scanners

Dynamic application security testing involves analyzing running applications. This methodology applies mainly to web applications and services and is used to find run-time vulnerabilities and environment-related issues. DAST scanners are run in later pipeline stages prior to deployment.

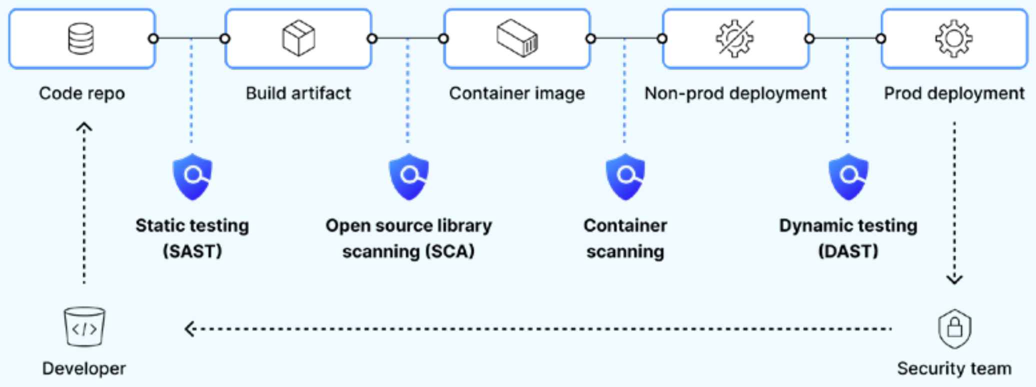


Figure 1: Orchestration of ASTs in a software development pipeline

Secure The Software Supply Chain

Achieving robust application security requires that DevSecOps take into account the entire software supply chain, which is defined as the sum total of all the code, people, systems, and processes that contribute to development and delivery of software artifacts, both inside and outside of an organization. Software supply chains are increasingly attractive threat vectors for attackers, and they are very hard to defend. Look no further than some of the major recent breaches such as Solarwinds, Log4j, and CodeCov to understand how critical it is to be able to identify and address security vulnerabilities in open source software artifacts and ensure the overall integrity of an application for customers and users.

With that said, let’s take a look at the critical capabilities DevSecOps teams are responsible for achieving.

Govern The Use Of Open Source Software Artifacts

Open-source software (OSS) has proliferated to the point where OSS artifacts are used in an overwhelming majority of applications. With this widespread use comes significant security and compliance risks. Each OSS component brings potential vulnerabilities into the applications that incorporate it, and it is incumbent upon every software producing organization to not use these artifacts blindly, but to properly vet them and govern their use throughout development pipelines. DevSecOps needs to track the deployment status of all artifacts and provide a real time view to customers to understand what artifacts are being used and where they are deployed. Furthermore, clear licensing policies and guidelines should be in place to ensure that users can confidently use the OSS artifact without infringing upon any legal requirements.

In the context of CI and CD pipelines, the best approach to establishing open source governance is through policy-as-code based on the Open Policy Agent (OPA) standard.

A good start to policy-based open source governance is to define rules for allowing or denying the use of OSS components based on criteria such as supplier, version, PURL (package URL), and license.

DevSecOps teams may create policies to block the use of a specific version of a component because it has known vulnerabilities or does not meet their organization's security standards. In this case, they can add that version of the component to the deny list, and any attempts to use it within the organization will be blocked.

Additionally, customers may want to block components from specific suppliers if they have had a history of security issues or if they do not meet their organization's compliance requirements. By specifying the supplier in the deny list, the customer can prevent any components from that supplier from being used within their organization. Or customers may want to block a component if it's not coming from a specific supplier.

The deny list is an important tool for customers to ensure the security and compliance of their organization by preventing the use of components that do not meet their standards.

Similarly, DevSecOps teams can define policies to allow the use of specific OSS artifacts that are approved by virtue of the fact they come from trusted suppliers, are contained in specific package types, and have safe licenses.

By enforcing these types of 'deny' and 'allow' rules, organizations can reduce the risk of security vulnerabilities, ensure compliance with licensing requirements, and maintain control over their software supply chain.

Ensure Software Integrity Through SLSA Compliance

Given the growing security risks inherent to today's software supply chains, it is increasingly important to software consumers to be able to trust the software they procure and use. But how would users and consumers know that a piece of software is indeed trustworthy? In determining the trustworthiness of a software artifact, you'd want to know about things like, who wrote the code? Who built it? Where (on which development platform) was it built? What components are in it?

Making a decision whether to trust software is possible once provenance– the record of a software's origins and chain of custody– can be verified. SLSA gives software-producing organizations the ability to capture information about any aspect of the SW chain, to verify properties of artifacts and their build, and to reduce risk of security issues.

In practice, it is essential for software producing organizations to adopt and adhere to the SLSA framework requirements and implement a means of verifying and generating software attestations which are authenticated statements (metadata) about software artifacts throughout their software supply chains.

The Supply Chain Levels for Software Artifacts (SLSA) Framework

Supply-chain Levels for Software Artifacts (SLSA) is a set of incrementally adoptable guidelines for supply chain security, established by industry consensus. The specification set by SLSA is useful for both software producers and consumers: producers can follow SLSA's guidelines to make their software supply chain more secure, and consumers can use SLSA to make decisions about whether to trust a software package.The SLSA framework offers:

- **A common vocabulary to talk about software supply chain security**
- **A way to secure your incoming supply chain by evaluating the trustworthiness of the artifacts you consume**
- **An actionable checklist to improve your own software's security**
- **A way to measure your efforts toward compliance with Executive Order 14028 standards**

Produce Comprehensive, Detailed Software Bill of Materials (SBOMs)

An SBOM is a detailed, machine-readable inventory of all libraries, modules, and dependencies involved in building a software artifact. It offers valuable transparency into the software's anatomy, ensuring both traceability and security. According to the National Telecommunications and Information Administration, a software bill of materials should document the following parameters:

- **Supplier Name**
  The name of an entity that creates, defines, and identifies components
- **Component Name**
  Designation assigned to a unit of software defined by the original supplier
- **Version of the Component**
  Identifier used by the supplier to specify a change in software from a previously identified version
- **Other Unique Identifiers**
  Other identifiers that are used to identify a component, or serve as a look-up key for relevant databases
- **Dependency Relationship**
  Characterizing the relationship that an upstream component X is included in software Y
- **Author of SBOM Data**
  The name of the entity that creates the SBOM data for this component
- **Timestamp**
  Record of the date and time of the SBOM data assembly

*source: https://www.ntia.doc.gov/files/ntia/publications/sbom_minimum_elements_report.pdf

By having an accurate and up-to-date SBOM, organizations can effectively manage and mitigate potential security risks. It enables them to identify and track vulnerable components, apply necessary patches and updates, and respond quickly to emerging threats. Additionally, an SBOM facilitates transparency and accountability by providing visibility into the software supply chain, allowing organizations to make informed decisions about the software they use and distribute.

Different industry standards have emerged for creating SBOMs, with CycloneDX and Software Package Data Exchange (SPDX) being the most prominent. CycloneDX is a lightweight software bill of materials (SBOM) standard designed by OWASP for use in application security contexts and supply chain component analysis. Software Package Data Exchange (SPDX) is an open standard by The Linux Foundation for communicating software bill of materials information, including provenance, license, security, and other related information. The significance of SBOMs has been further underlined by Executive Order 14028, pushing it to the forefront of cybersecurity discussions.

Executive Order 14028

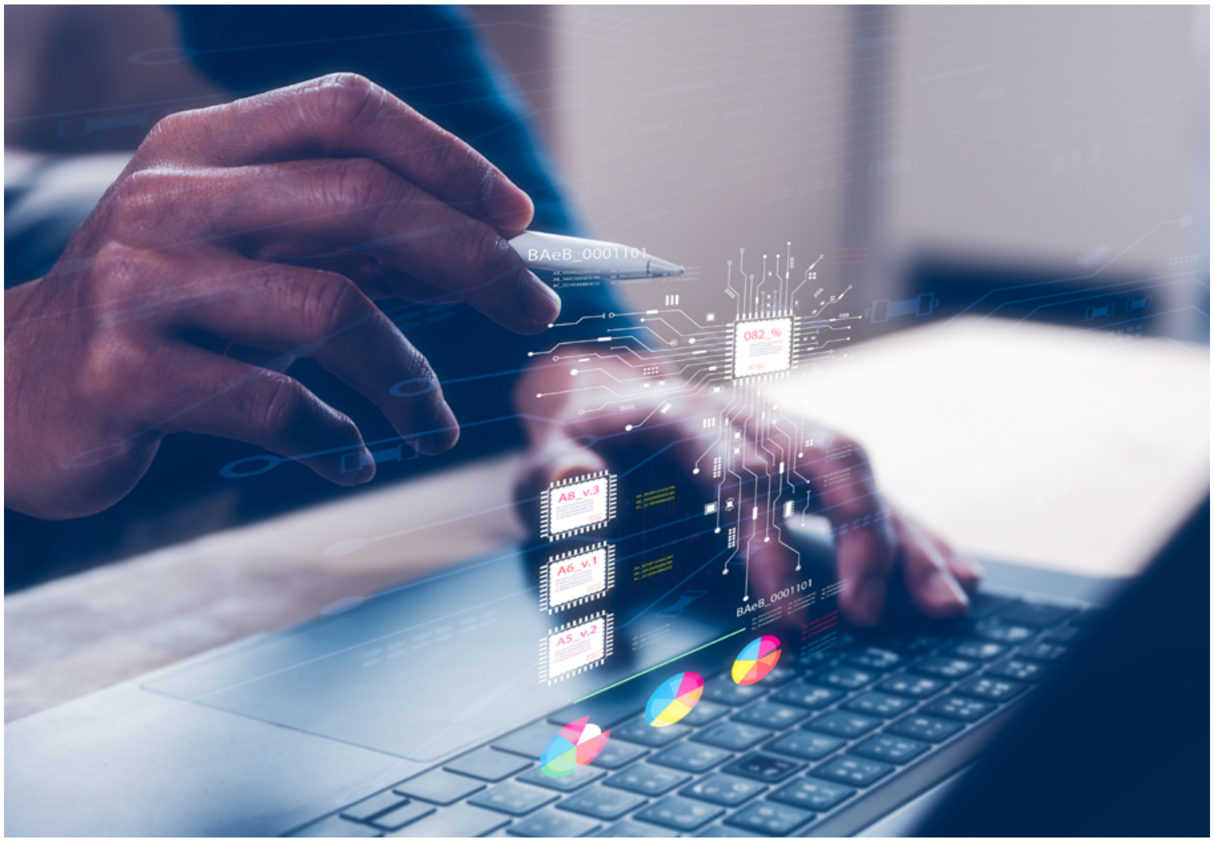
Issued by The White House in May, 2021, Executive Order 14028 aims to bolster the nation's cybersecurity infrastructure. A key focus of this order is enhancing software supply chain security, and SBOMs have been highlighted as a critical tool in this context. The order has accelerated the adoption and standardization of SBOMs, pushing organizations to implement robust SBOM management practices.

The Value of SBOMs

- **Transparency:** Provides full visibility into the components in use and their metadata
- **Compliance:** Ensures adherence to Executing Order 14028 and to licensing requirements
- **Security:** Enables swift response to new vulnerabilities by precisely identifying impacted components

Making SBOM Management Part of Your DevSecOps Practice

SBOMs have a life cycle of their own, throughout which they need to be generated, updated, verified, and signed. Typically, an SBOM is first generated within the build phase, where much of the critical data for a reliable detailed SBOM exists. Updates can happen frequently; for example, scanning software dependencies recursively after a build may require changes to the SBOM. Ultimately, lifecycle management needs to be automated and easily orchestrated across software development pipelines without slowing down developers as they build their applications.



How Harness Provides The Foundation For Effective DevSecOps

Harness takes a holistic, platform-based approach to securing CI/CD pipelines along with the software supply chain that enhances collaboration, governance and control, and developer productivity. Powered by two fully-integrated Harness Platform modules (Security Test Orchestration (STO) and Software Supply Chain Assurance (SSCA)), Harness DevSecOps enables you to orchestrate vulnerability scanners anywhere in your pipeline, intelligently remediate vulnerabilities quickly using AI, govern OSS components with detailed SBOM, and ensure artifact integrity for SLSA compliance.

Securing Applications with Harness Security Testing Orchestration (STO)

Harness makes security testing orchestration flexible and simple, allowing organizations to step away from bolted-on point solutions and put an end to the flood of disparate vulnerability data sets that complicate remediation efforts. Instead, Harness streamlines vulnerability detection, deduplication, prioritization, and remediation in ways that minimize both security risk and developer toil. Developers benefit from AI-driven remediation assistance, along with security exemption management that mitigates unnecessary deployment roadblocks.

Establish OSS visibility and Governance

Modern applications are built with a myriad of open source software (OSS) artifacts and 3rd party components which introduce new vulnerabilities that put both software vendors and consumers at risk. Tracking the usage of OSS components across software artifacts and environments and enforcing policies governing their use is an essential DevSecOps capability that relies on a detailed software bill of materials (SBOM). The SBOM is essential for understanding the components and dependencies within an application, which in turn enables your organization to manage open-source component risks effectively. In the event a zero-day vulnerability is discovered, Harness SSCA enables rapid remediation by tracing the vulnerability back to impacted artifacts.

Ensure Software Artifact Integrity

Software supply chain attacks are becoming more aggressive, sophisticated, and diverse. In the wake of high profile breaches such as Solarwinds and Log4j, there is substantial pressure on organizations to implement safe development practices and ensure the integrity of their applications and all of their artifacts. Harness SSCA brings your DevSecOps practices into alignment with Supply Chain Levels for Software Artifacts (SLSA) Level-2 and enables you to meet the requirements of Executive Order 14028.

Harness Secure Software Delivery Capabilities at a Glance

Harness Secure Software Delivery Capabilities at a Glance

- Seamless integration with over 40 commercial and open source security scanners
- Orchestration of application security tests anywhere in your pipelines
- Policy-as-Code governance of the security pipeline with OPA
- Automatic deduplication and prioritization of vulnerabilities
- AI-driven vulnerability remediation guidance
- Generation and attestation of SLSA Level-3 provenance with Harness CI hosted builds
- Generation and attestation of SBOMs
- SLSA Level-2 provenance verification
- Rapid remediation of zero-day vulnerabilities

harness

The Modern Software Delivery Platform™

Follow us on

/harnessio

/harnessinc

Contact us on

www.harness.io