4 BEST PRACTICES FOR PROTECTING AGAINST SOFTWARE SUPPLY CHAIN ATTACKS

digicert®

Integrating secure code signing, threat and malware detection, and SBOM generation

Software supply chain attacks are increasing in frequency. A recent Gartner report says that 45% of companies worldwide will experience attacks on their software supply chains by 2025, a threefold jump from 2021.

Companies large and small are experiencing these attacks. By now, most everyone has heard of supply chain attacks at SolarWinds, CircleCI and 3CX. More recently, Micro-Star International (MSI) suffered an attack that compromised the private code signing keys used for its BIOS as well as keys used for Intel's BootGuard.

While basic security measures, such as Zero Trust principles, provide companies with a first line of defense, more action is needed to secure a company's software supply chain and software development lifecycle.

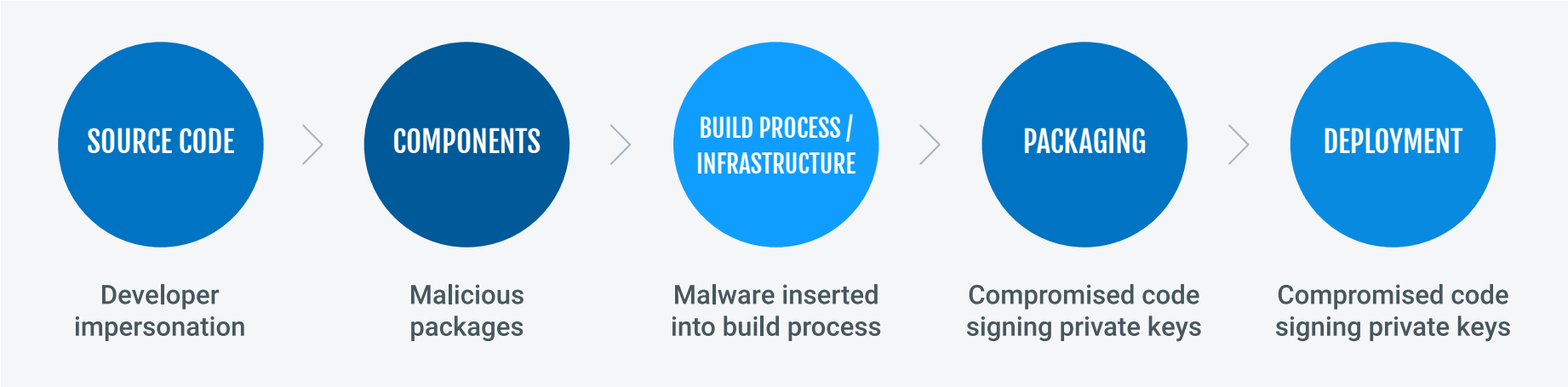
WHO SHOULD READ THIS:

Security, DevSecOps, and software engineering professionals who are responsible for, or have a stake in, protecting their company's software development infrastructure from breaches, attacks, and other vulnerabilities.

CHALLENGES

Malicious actors use a variety of tactics when attacking a software supply chain. They might target a compromised software developer's password to break into a software build system, steal or misuse unprotected code signing private keys, insert malware upstream in common open-source software packages, or use a combination of all these tactics. As the Sunburst attack on SolarWinds demonstrated in 2019, attack tactics are becoming quite sophisticated and complex, which means that more sophisticated methods of fighting these attacks are required, as shown in Figure 1.

Figure 1: Supply chain attacks can happen at any stage



Attacks can occur during any stage of the software development lifecycle. Stand-alone existing security tooling might detect specific types of attacks but will miss other types. Therefore, a "shift up" approach is necessary to add security measures throughout the software development lifecycle.

Other challenges are organizational and cultural: Software development teams may resist security tools because they can be cumbersome, break automated CI/CD build pipelines or slow down the software release process.

Security teams have a different set of challenges. They need to provide security guidance and enforce compliance with many software teams spread across the organization, often geographically distributed. In addition, these teams often use different programming languages, and environments, deploy software to different platforms and use different software build infrastructure and software processes. This heterogeneity can make defining, enforcing and monitoring software security policy across an enterprise extremely difficult.

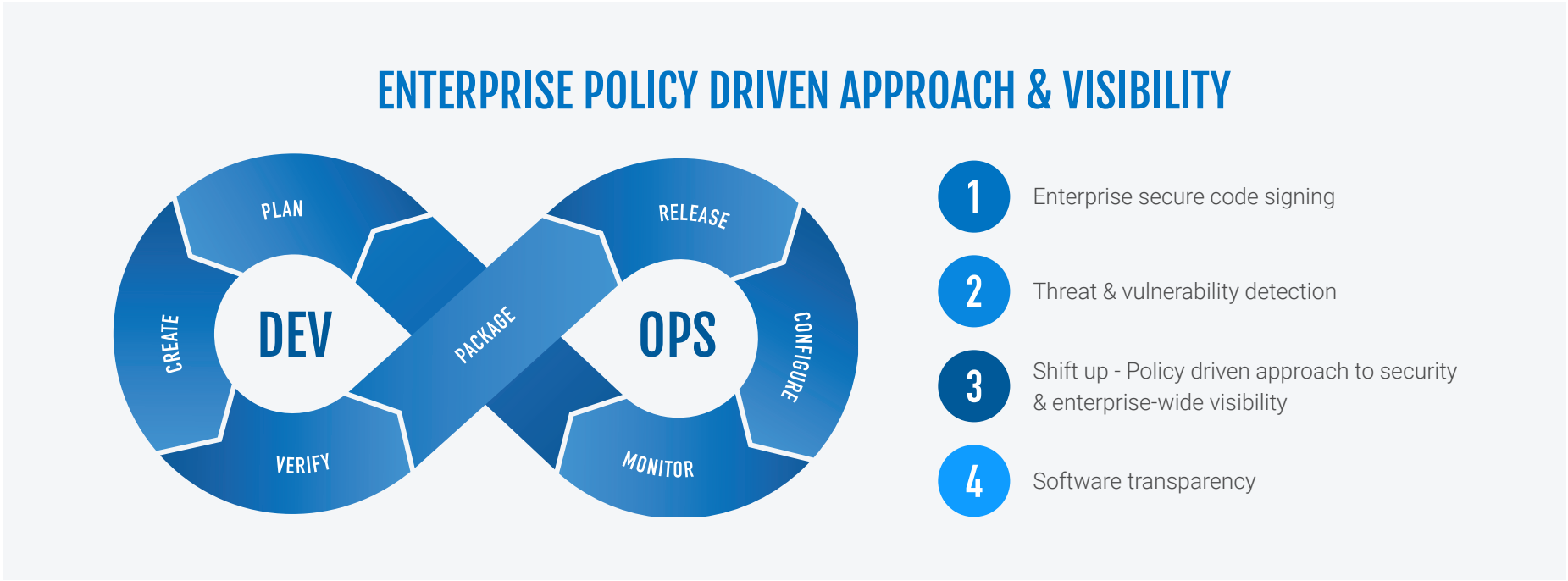
Finally, a new challenge for organizations is compliance with various government and industry regulations that are meant to address software supply chain attacks, such as those that require software transparency.

BEST PRACTICES FOR SECURING SOFTWARE SUPPLY CHAINS

Several regulations and industry recommendations specify not only best practices, but requirements that companies must follow to ensure the integrity of the software supply chain. For example, the US Executive Order on Improving the Nation's Cybersecurity and the EU's Cyber Resilience Act requires that software bills of materials be generated to show software transparency. NIST's Security Considerations for Code Signing outlines steps that should be followed for secure code signing.

This solution brief outlines four best practices that companies can leverage to address these challenges, along with the ways DigiCert can help you. The key aspect of these best practices is that they occur throughout the software development lifecycle instead of focusing on a single SDLC stage (see Figure 2).

Figure 2: Software Trust: shifting left, right... AND UP



A multi-faceted, integrated, next-gen approach is needed to fight software supply chain attacks focused on multiple aspects of the SDLC.

#1: Secure Code Signing Across the Enterprise

Code signing is a basic security measure that has been in use for over 30 years. It uses cryptography to digitally sign a software package (executable, library, app, etc.) to show that the software is authentic (e.g., software signed by DigiCert really comes from DigiCert) and has not been tampered with by a third party. Many operating systems, like iOS, macOS or Windows, will check the digital signature of software before installing and running the package.

Software signing is so effective that malicious actors have been targeting code signing systems to steal, compromise or misuse a company's code signing certificates and private keys.

An obvious first step in securing code signing is ensuring that all private code signing keys are securely stored. A recent requirement from the Certificate Authority/Browser Forum requires the storage of publicly trusted private code signing keys in a FIPS 140 Level 2, Common Criteria EAL 4+ device such as a hardware security module (HSM).

While using secure storage is a necessary first step, it is not sufficient to secure code signing across an enterprise. The reason for this is that a compromised username/password could allow a malicious actor to access the secure storage.

Instead, we recommend the use of an enterprise-strength code signing system that offers a number of additional security measures, including:

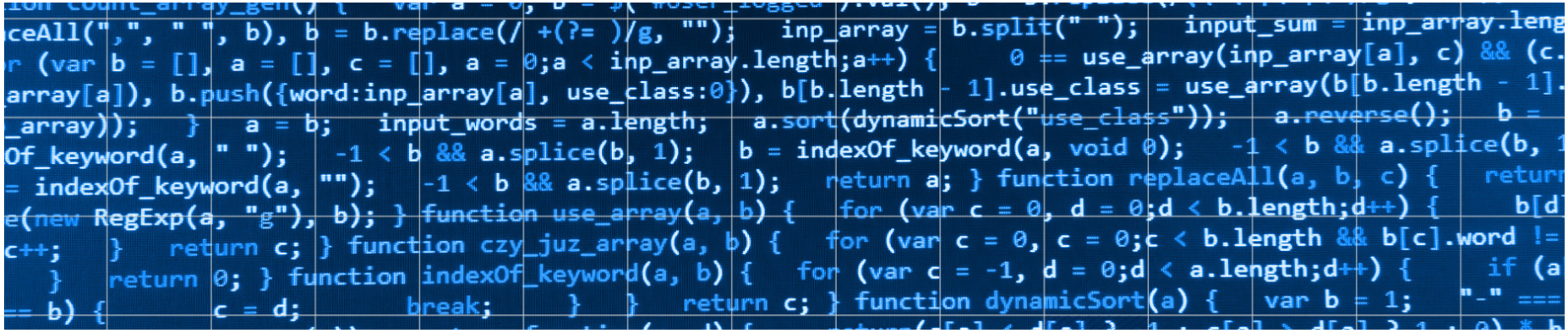
- Automation of code signing certificate management, including issuance and revocation.
- An approval process that specifies not only who can access a code signing key but who needs to approve the access and under what circumstances the key can be accessed (such as time of day, build machine, etc.).
- A role-based system that specifies which users can access certain code signing certificates and keys for signing operations, which are authorized to approve signing operations, and which are allowed to monitor.
- Maintaining an irrefutable record of all code signing activities to refer to in the event of a compromised certificate.
- A single code signing system that supports a large volume of code signing operations and many different software deployment and development platforms and tools.
- A policy-based approach of control measures that must be satisfied before software can be released.
- Sign intermediate software artifacts throughout the SDLC, including source code before it is committed, build scripts and recipes, and container and other software infrastructure files.
- Enterprise-wide visibility, policy definition and enforcement from a centralized location no matter where various software teams are located.

#2: Comprehensive Threat Detection

While code signing can prevent tampering after software is released, what happens if software is tampered with before it is signed and released? For example, a malicious actor with a breached build system could insert malware into a company's source code repository. Or a software developer could download an open-source package that has hidden malware in it. Or a previously undetected hidden vulnerability could be discovered.

Various security tools on the market address some of these issues, such as software static analysis tools (SAST). However, these tools often only focus on a single aspect (such as vulnerabilities with open-source software).

Instead, a comprehensive method is needed to look at scan final software binaries for threats, vulnerabilities, presence of embedded secrets and malware. This scan should occur no matter the original source of the components of the software: open-source code, third-party commercial libraries or proprietary code written by the company.



#3: Software Transparency

As mentioned earlier, government and industry regulations now more frequently require software publishers to reveal what's in their software, much like a nutrition label indicates the ingredients in food. Known as software bills of materials (SBOMs), these documents catalog all the components in a given piece of software no matter the component's source (e.g., open-source, third-party library, etc.). As with a food nutrition label, SBOMs not only indicate a list of software "ingredients" but can also be used to assess the quality of those ingredients, such as versions used, known vulnerabilities, etc.

Even if not required for compliance, organizations should generate SBOMs so they can:

- **Evaluate and prioritize the attack surface of their software**: Look for vulnerable or targeted versions of libraries or components, look at software dependencies and check to see if there are missing mandated security patches.
- **Look for outdated components** like missing mitigations and insecure code signing practices.
- **Take Proactive Defensive Measures** by using software structure to support detection and threat incident response teams and for threat modeling.

#4: Shift Up: Control and Visibility Across the Enterprise and SDLC

Many security measures look at/protect a specific part of the SDLC. We've heard about shifting security left and shifting security right in a DevOps process, which is akin to looking at individual trees in a forest. But as attacks become more complex, organizations need to step back and shift their focus up:

- Bring the entire SDLC process into view, looking at the software as a whole.
- Have visibility of security policy, measures and results across the entire enterprise.
- Be able to define and enforce security policy across the entire enterprise.
- Perform threat and vulnerability analysis on final software executables immediately before they are signed and released.

To effectively do this, as well as to be accepted by software development teams, security tools must integrate seamlessly into existing software processes and not slow down the software release process. They need to be easy, or even transparent, for software developers to use.

DIGICERT SOFTWARE TRUST MANAGER

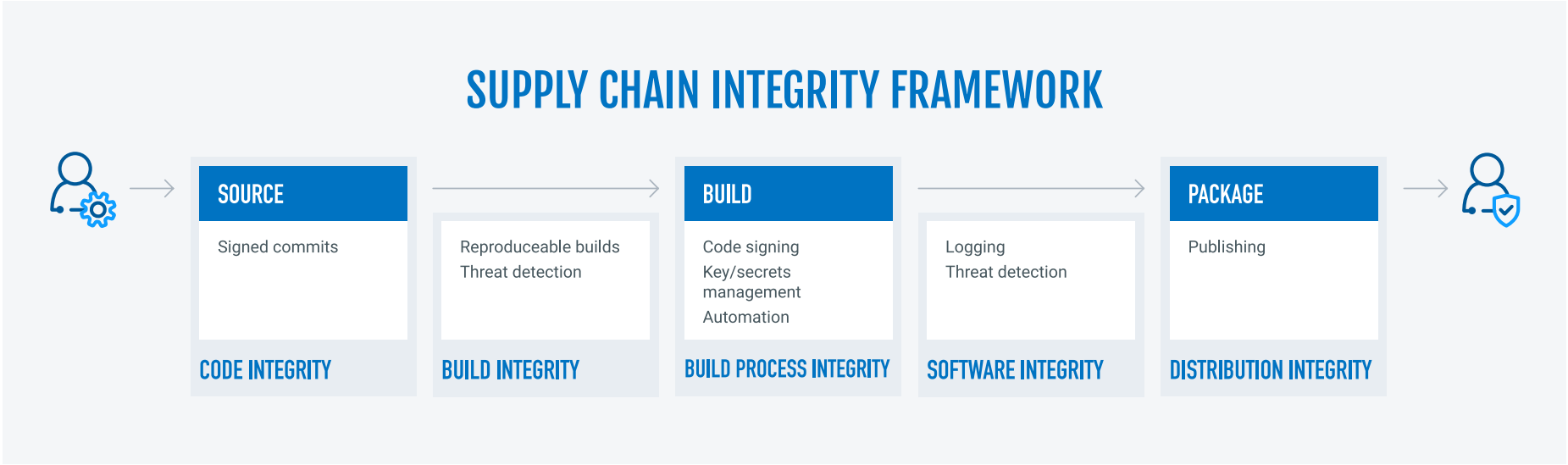
DigiCert® Software Trust Manager is a digital trust solution that protects the integrity of software across the software supply chain, reducing risk of code compromise, enforcing corporate and regulatory policy, and delivering fine-grained key usage and access controls for code signing.

DigiCert Software Trust Manager minimizes the risks of security breach and malware propagation in the development, build and release of software, enabling companies to protect against software supply chain attacks and comply with government regulations.

DigiCert Software Trust Manager helps large enterprises easily implement these four best practices.

To find out more, please email pki_info@digicert.com.

Figure 3: Software Trust Manager



DigiCert Software Trust Manager protects across the software development lifecycle by providing secure code signing, comprehensive threat detection, software transparency, and enterprise visibility and control.