SUPPLY CHAIN LOGICIELLE : 4 BONNES PRATIQUES DE PROTECTION CONTRE LES ATTAQUES

digicert®

Comment intégrer la signature de code, la détection des menaces et des malwares, et la création d'une SBOM

La supply chain logicielle figure de plus en plus dans le viseur des attaquants. Selon un rapport Gartner récent, 45 % des entreprises à travers le monde subiront des attaques contre leur supply chain logicielle d'ici 2025, soit trois fois plus qu'en 2021.

Et aucune entreprise n'est à l'abri, qu'elle soit petite ou grande. Parmi ces attaques de grande envergure, citons celles visant SolarWinds, CircleCI et 3CX. Plus récemment encore, l'entreprise Micro-Star International (MSI) a été victime d'une attaque compromettant les clés privées de signature de code associées à son BIOS et les clés utilisées pour sa protection Intel BootGuard.

Si certaines mesures de sécurité élémentaires, telles que les principes Zero Trust, assurent aux entreprises une première ligne de défense, d'autres actions s'imposent pour sécuriser à la fois la supply chain et le cycle de développement logiciels d'une entreprise.

DESTINATAIRES DE CE GUIDE :

Professionnels de la sécurité, équipes DevSecOps et ingénieurs logiciels chargés de protéger (ou impliqués dans la protection de) l'infrastructure de développement logiciel de leur entreprise contre les compromissions, attaques et autres vulnérabilités.

PROBLÉMATIQUES

Les cybercriminels recourent à une variété de techniques pour attaquer une supply chain logicielle : utilisation d'un mot de passe compromis pour s'infiltrer dans un système de développement logiciel, vol ou mauvaise utilisation de clés privées non protégées pour la signature de code, introduction de malware dans des logiciels open-source courants, ou combinaison de toutes les méthodes précitées. À l'instar de l'assaut Sunburst sur SolarWinds en 2019, les modes opératoires deviennent de plus en plus complexes et sophistiqués. Pour les contrer, il faut donc employer des systèmes de défense tout aussi élaborés (voir Figure 1).

Figure 1. Chaque maillon de la supply chain est menacé

Les attaques peuvent survenir à n'importe quel stade du développement logiciel. Les outils de sécurité spécialisés existants peuvent détecter certains types d'attaques, mais pas tous. D'où l'importance d'une approche « shift up » qui consiste à intégrer des mesures de sécurité tout au long du développement logiciel.

Les autres défis à relever sont d'ordre organisationnel et culturel. Lorsque les équipes de développement logiciel se montrent réticentes à l'égard d'outils de sécurité, c'est parce que ceux-ci peuvent être contraignants, interrompre les pipelines CI/CD automatisés ou encore ralentir le processus de déploiement des logiciels.

Pour les équipes de sécurité, l'enjeu est d'une tout autre nature. Elles doivent apporter des conseils de sécurité et assurer la conformité dans l'ensemble des équipes logicielles de l'entreprise, bien souvent réparties à travers le monde. Pour compliquer encore plus la tâche, ces équipes ont recours à différents langages de programmation et différents environnements. Elles déploient des logiciels sur diverses plateformes et exploitent une variété d'infrastructures et de processus de développement logiciel. Face à cette hétérogénéité, la définition, l'application et le suivi des politiques de sécurité logicielle à tous les niveaux de l'entreprise relèvent de la gageure.

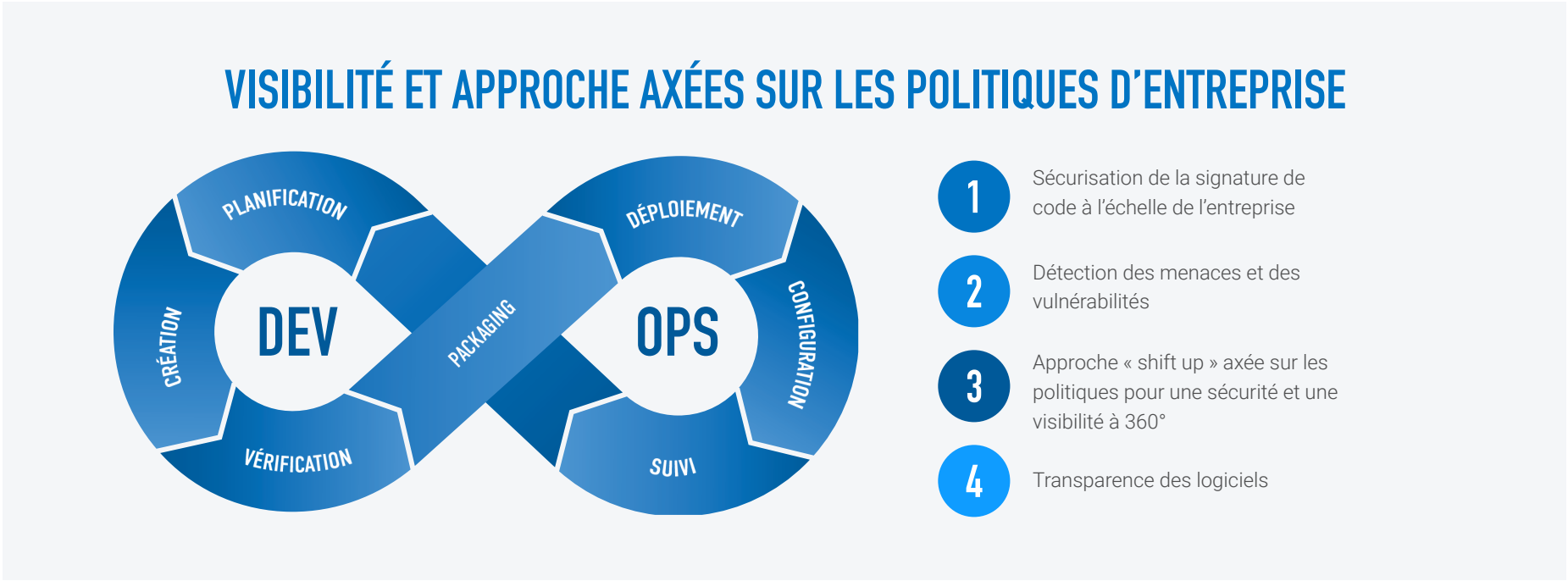
Enfin, dernière problématique à prendre en compte : la conformité des entreprises à l'égard des multiples réglementations gouvernementales et sectorielles, pensées pour prévenir les attaques contre la supply chain logicielle. C'est le cas, par exemple, des dispositions sur la transparence des logiciels.

BONNES PRATIQUES DE SÉCURISATION DES SUPPLY CHAINS LOGICIELLES

Pour assurer l'intégrité de la supply chain logicielle, plusieurs réglementations et recommandations sectorielles énoncent non seulement de bonnes pratiques, mais aussi des exigences auxquelles les entreprises doivent se plier. Parmi elles, l'ordre exécutif des États-Unis visant à renforcer la cybersécurité des institutions du pays et le règlement de l'UE sur la cyberrésilience, qui exigent de générer une nomenclature des composants logiciels (SBOM) pour accroître la transparence des logiciels. Le NIST a également publié des considérations de sécurité pour la signature de code qui détaillent toutes les étapes à suivre pour sécuriser le processus de signature.

Quant à notre guide, il propose quatre bonnes pratiques que les entreprises peuvent adopter pour résoudre ces problématiques et met en lumière les solutions apportées par DigiCert. Ces bonnes pratiques sont conçues pour s'implémenter tout au long du développement logiciel et non pas simplement à une étape particulière du cycle (voir Figure 2).

Figure 2. Confiance pour les logiciels : approches « shift left », « shift right »… et « SHIFT UP »



Implémentée tout au long du cycle de développement, une nouvelle approche intégrée et multifacette s'impose pour bloquer les attaques contre la supply chain logicielle.

N° 1 – Sécurisation de la signature de code à l'échelle de l'entreprise

La signature de code est une mesure de sécurité élémentaire qui fait ses preuves depuis plus de 30 ans. Basé sur la cryptographie, ce processus permet de signer numériquement un package logiciel (fichier exécutable, bibliothèque, application, etc.) pour authentifier un logiciel (ex. : pour prouver qu'un logiciel signé par DigiCert provient bel et bien de DigiCert) et garantir son intégrité. De nombreux systèmes d'exploitation, comme iOS, macOS ou Windows, vérifient la signature numérique des logiciels avant leur installation et leur exécution.

La signature de logiciels est tellement efficace que les hackers s'en prennent aux systèmes de signature de code pour voler, compromettre ou utiliser à mauvais escient les certificats de signature de code et les clés privées d'une entreprise.

Pour sécuriser ce processus, l'une des premières étapes consiste évidemment à s'assurer que toutes les clés privées de signature de code sont bien à l'abri. Conformément à une exigence récente du CA/B Forum (Certification Authority/ Browser Forum), les clés privées de signature de code utilisées par le grand public doivent être stockées sur un appareil certifié FIPS 140 niveau 2, critères communs EAL 4+, tel qu'un module matériel de sécurité (HSM).

Si la sécurisation du stockage est une première étape obligatoire, cela ne suffit toutefois pas à protéger la signature de code dans l'ensemble de l'entreprise. Pourquoi ? Car la simple compromission d'un nom d'utilisateur/mot de passe permettrait à un cybermalfaiteur d'accéder au stockage sécurisé.

Nous recommandons donc de recourir plutôt à un système de signature de code applicable à l'ensemble de l'entreprise. Parmi les fonctionnalités de sécurité recherchées dans cet outil complet :

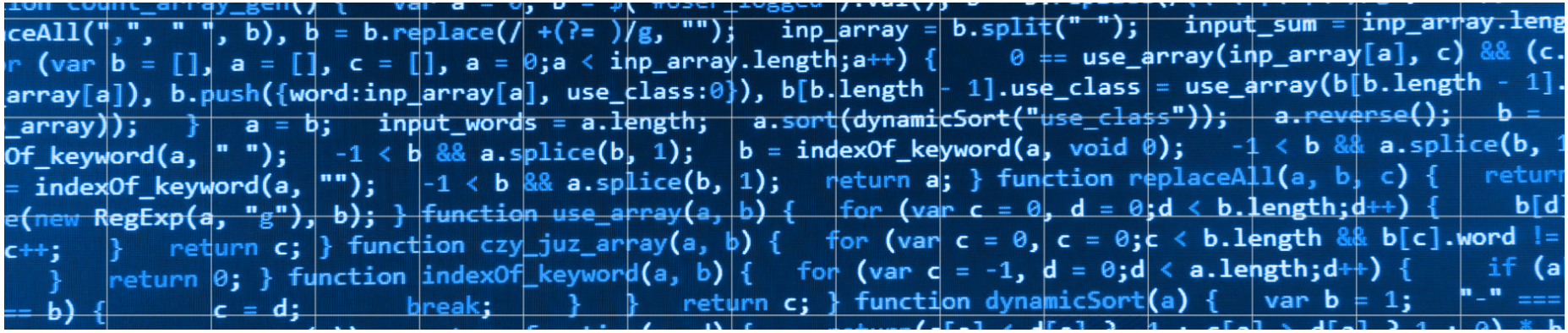
- Automatisation de la gestion des certificats de signature de code, y compris de leur émission et de leur révocation
- Processus d'approbation qui détermine non seulement les utilisateurs autorisés à accéder à la clé de signature de code, mais aussi les utilisateurs qui doivent approuver ces accès et les circonstances dans lesquelles la clé peut être utilisée (horaires, machines, etc.)
- Système basé sur les rôles qui spécifie 1) les utilisateurs pouvant accéder à certains certificats de signature de code et à certaines clés pour les opérations de signature ; 2) les utilisateurs autorisés à approuver les opérations de signature ; et 3) les utilisateurs autorisés à superviser le processus
- Enregistrement systématique et incontestable de toutes les activités de signature de code pour référence en cas de compromission d'un certificat
- Système unique de signature de code qui prend en charge un grand volume d'opérations de signature, de nombreux déploiements logiciels et divers outils et plateformes de développement
- Mesures de contrôle basées sur des politiques à appliquer aux logiciels avant leur déploiement
- Signature des artefacts logiciels intermédiaires tout au long du cycle de développement, y compris du code source avant tout commit, des scripts et recipes de build, des containers et des autres fichiers liés à l'infrastructure logicielle
- Plateforme centralisée pour une visibilité, une définition des politiques et une mise en application à l'échelle de l'entreprise, peu importe où se trouvent les différentes équipes logicielles

N° 2 – Détection des menaces d'un bout à l'autre du cycle logiciel

Si la signature de code protège l'intégrité du logiciel une fois déployé, comment faire pour empêcher qu'un logiciel soit altéré avant sa signature et son déploiement ? Il peut s'agir, par exemple, d'un hacker qui, en exploitant un système de développement compromis, parvient à introduire un malware dans le référentiel de code source d'une entreprise. Ou encore d'un développeur de logiciels qui télécharge un package open-source renfermant un malware caché, ou bien d'une vulnérabilité jusque-là dissimulée qui vient d'être découverte.

Divers outils de sécurité déjà disponibles aident à résoudre certains de ces problèmes, tels que les outils SAST (tests statiques de la sécurité applicative). Seulement voilà, ces outils tendent à ne se concentrer que sur un seul aspect (tel que les vulnérabilités dans les logiciels open-source).

Ce dont les entreprises ont besoin, c'est d'une méthode complète d'analyse des fichiers binaires finaux des logiciels pour détecter à la fois les menaces, les vulnérabilités, la présence de secrets cachés et les malwares. Cette analyse doit s'effectuer, quelle que soit l'origine des composants du logiciel : code open-source, bibliothèques commerciales tierces ou code propriétaire créé par l'entreprise.



N° 3 – Transparence des logiciels

Comme évoqué précédemment, les réglementations gouvernementales et sectorielles exigent de plus en plus des éditeurs logiciels qu'ils révèlent le contenu de leurs programmes, à l'image des étiquettes nutritionnelles qui indiquent, entre autres, les ingrédients dont sont composés les aliments. Appelés nomenclatures des composants logiciels (SBOM), ces documents référencient tous les composants d'un logiciel donné, quelle que soit son origine (open-source, bibliothèque tierce, etc.). À l'instar des étiquettes nutritionnelles, les SBOM ne se contentent pas d'énumérer une liste « d'ingrédients » logiciels. Ellles peuvent également servir à évaluer la qualité de ces ingrédients, tels que les versions utilisées, les vulnérabilités connues, etc.

Bien qu'elles ne soient pas obligatoires, ces nomenclatures s'avèrent indispensables aux entreprises pour :

- **Évaluer et prioriser les menaces qui pèsent sur leur logiciel** en identifiant les versions vulnérables ou ciblées des bibliothèques ou composants, en inspectant les dépendances logicielles et en vérifiant l'application de tous les correctifs de sécurité nécessaires
- **Identifier les éléments obsolètes** tels que les mesures de prévention à actualiser ou les pratiques non sécurisées de signature de code
- **Adopter des lignes de défense préventives** basées sur la structure des logiciels pour aider les équipes de détection et de réponse à incident, et à des fins de modélisation des menaces

N° 4 – Approche « shift up » pour un contrôle et une visibilité à 360°

De nombreuses solutions de sécurité visent à analyser et protéger une part spécifique du cycle de développement logiciel. Pour les DevOps, les approches « shift left » et « shift right » de la sécurité sont bien connues et consistent, en quelque sorte, à observer chacun des arbres d'une forêt. Mais au vu de la complexité croissante des attaques, les entreprises doivent prendre de la hauteur et adopter une perspective globale dite « shift up » :

- Visualiser l'ensemble du processus de développement logiciel pour examiner le logiciel de A à Z
- Assurer une visibilité sur les politiques, mesures et résultats en matière de sécurité à tous les niveaux de l'entreprise
- Définir et appliquer les politiques de sécurité dans l'ensemble de l'entreprise
- Analyser les menaces et les vulnérabilités sur les fichiers exécutables finaux des logiciels juste avant leur signature et leur déploiement

Pour être efficaces et être acceptés par les équipes logicielles, les outils de sécurité doivent s'intégrer en toute fluidité aux processus existants, sans ralentir les phases de déploiement. C'est grâce à la simplicité d'utilisation, voire à la transparence, de ces outils que les développeurs de logiciels se les approprieront totalement.

DIGICERT SOFTWARE TRUST MANAGER

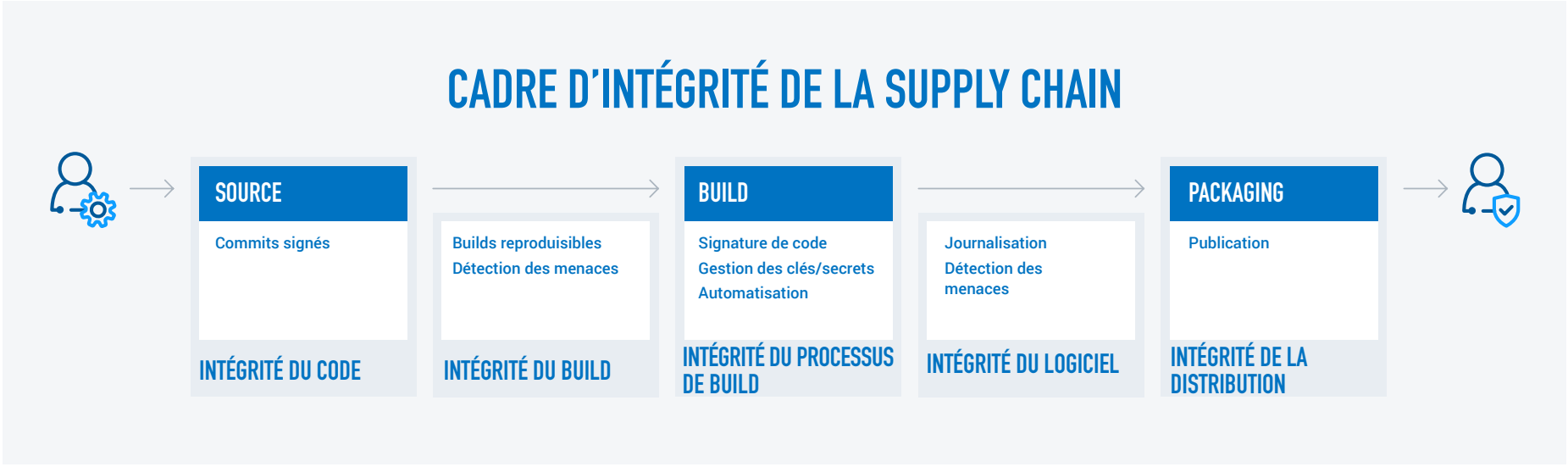
DigiCert® Software Trust Manager est une solution de confiance numérique qui garantit l'intégrité du code sur la supply chain logicielle. À la clé : réduction du risque de compromission du code, respect des réglementations, application des politiques internes, et granularité dans l'usage des clés et les contrôles d'accès pour la signature de code.

Software Trust Manager réduit les risques de compromission et de propagation des malwares dans les phases de développement, de build et de déploiement des logiciels. La solution permet donc aux entreprises de se prémunir contre les attaques sur leur supply chain logicielle tout en respectant les dispositions légales.

DigiCert Software Trust Manager aide les grandes entreprises à adopter ces quatre bonnes pratiques en toute simplicité.

Pour en savoir plus, écrivez à l'adresse pki_info@digicert.com.

Figure 3. Software Trust Manager



Signature de code sécurisée, détection complète des menaces, transparence logicielle, visibilité et contrôle à l'échelle de l'entreprise… DigiCert Software Trust Manager assure une protection d'un bout à l'autre du développement logiciel.