5 stratégies pour sécuriser votre supply chain logicielle

digicert

Zoom sur la signature de code, la détection des menaces et des malwares, et la création d'une SBOM (liste complète des composants logiciels)

Les incidents dans la supply chain logicielle ont augmenté en fréquence et en gravité depuis plusieurs années. Un rapport de TechTarget Enterprise Strategy Group a révélé que 91 % des organisations ont connu une telle situation en 2023.

Les entreprises, grandes et petites, subissent ces attaques. Aujourd'hui, tout le monde ou presque a entendu parler des attaques contre les supply chain de SolarWinds, CircleCI et 3CX. En mars 2024, le développeur d'un outil de compression Linux très répandu s'est fait piéger pour introduire une porte dérobée (backdoor) dans le code. Le problème a été détecté avant que la plupart des distributions de Linux les plus connues ne l'intègrent dans leurs versions de production, mais les conséquences auraient pu être terribles.

Si les mesures de sécurité de base, telles que le Zero Trust, constituent une première ligne de défense pour les entreprises, une approche de défense renforcée (DiD) est nécessaire pour sécuriser les supply chains et les cycles de développement logiciel complexes d'une entreprise.

Problématiques

Les acteurs malveillants utilisent diverses tactiques pour s'attaquer à une supply chain logicielle (voir figure 1). Ils peuvent se servir du mot de passe compromis d'un développeur pour s'introduire dans un système de build, voler ou faire un mauvais usage de clés privées de signature de code non protégées, insérer des malwares dans des packages open-source courants, voire utiliser une combinaison de toutes ces tactiques. Comme l'a démontré l'attaque Sunburst contre SolarWinds en 2019, les tactiques d'attaque peuvent être très sophistiquées et complexes, mais les défenses n'ont pas besoin d'un tel niveau de technicité pour être efficaces. Le respect des bonnes pratiques, des principes de défense en profondeur et l'adoption d'une approche cohérente de la sécurité sont à la portée de tous.

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 pour de nombreuses raisons : ceux-ci peuvent être contraignants,

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. Par conséquent, les organisations devraient adopter des stratégies « shift left » et « shift right » pour sécuriser l'ensemble du cycle de développement logiciel.

interrompre les pipelines CI/CD automatisés ou encore ralentir le processus de déploiement des logiciels. Les faux positifs sont fréquents, et même les vraies alertes peuvent donner l'impression que les produits de sécurité sont importuns et inutiles.

Pour les équipes de sécurité, l'enjeu est d'une tout autre nature. Elles doivent apporter des conseils de sécurité et assurer la conformité de nombreuses équipes logicielles dans l'entreprise, lesquelles étant bien souvent dispersées géographiquement et dans l'organigramme de l'entreprise. 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 grande variété d'infrastructures et d'applications. 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, la plupart des organisations, en particulier dans certaines régions comme l'UE, doivent se conformer à un nombre croissant de réglementations gouvernementales et de standards privés. 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, lles décrets américains visant à renforcer la cybersécurité des institutions du pays et le règlement de l'UE sur la cyber-résilience, qui exigent de générer une nomenclature des composants logiciels (SBOM) pour accroître leur transparence. 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 l'article 6 du standard Payment Card Industry Data Security Standard (PCI-DSS), il énonce plusieurs exigences pour le développement et la maintenance de logiciels et de systèmes sécurisés.

Ce document présente cinq stratégies, ou 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 3, page 3).

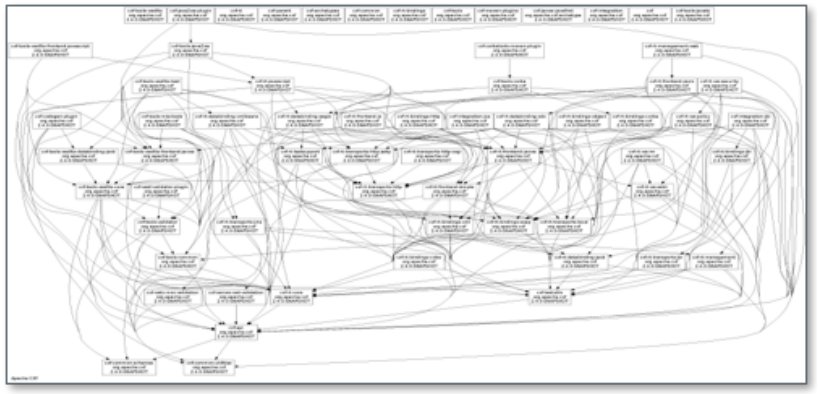
N° 1 – Ne sous-estimez pas la complexité de vos environnements de développement

Pourquoi tant d'attaques contre la supply chain logicielle atteignent-elles leur but ? Les raisons sont multiples : les équipes informatiques sont souvent en sous-effectif, isolées et en vase clos.

Une grande structure présente probablement une surface d'attaque vaste et complexe, ouverte à diverses attaques provenant du monde entier. Mais l'une des principales raisons est sans aucun doute la complexité des systèmes logiciels modernes.

La figure 2 montre une représentation structurelle du projet logiciel Apache HTTP en 2024. Lancé en 1995 par 8 développeurs, ce projet compte aujourd'hui environ 2 millions de lignes de code et plus de 630 000 contributeurs dans le monde.

Figure 2 : Projet logiciel Apache http

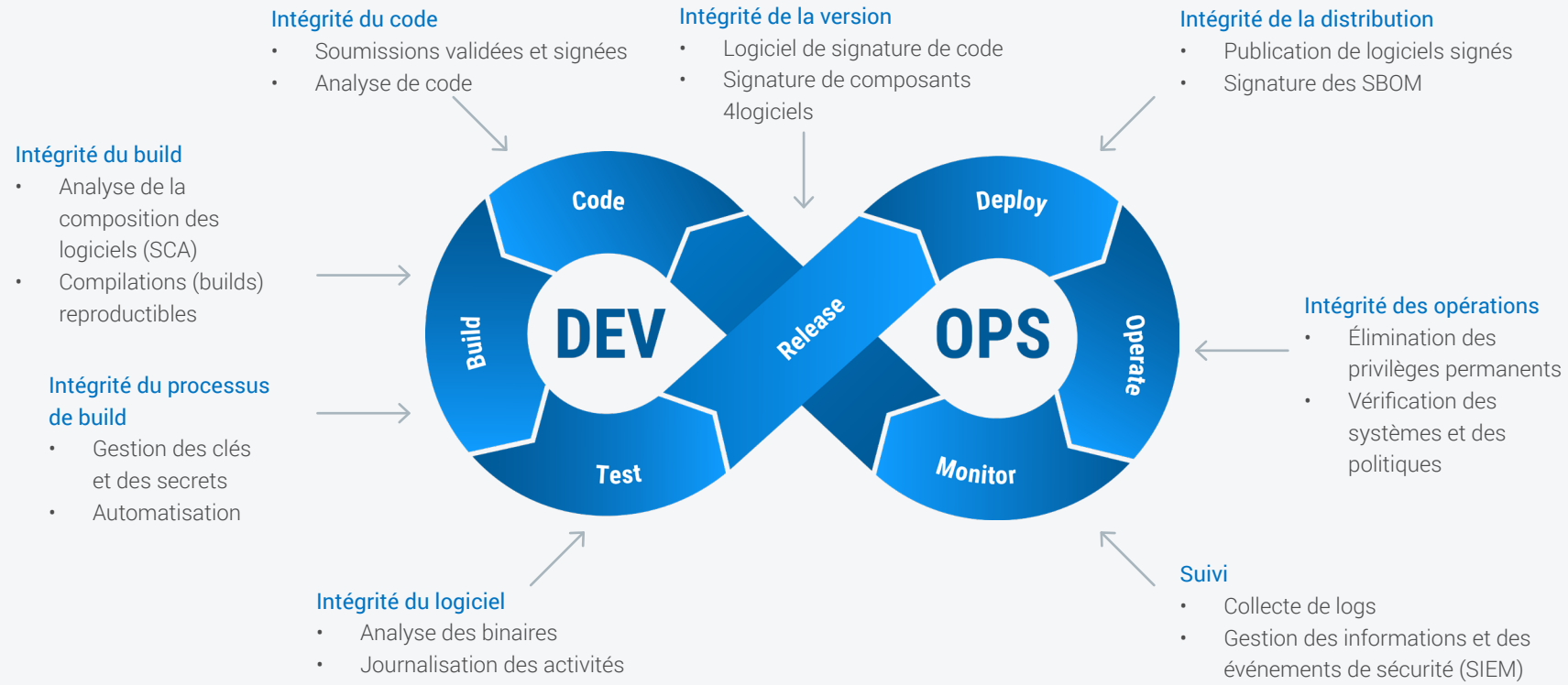


Apache HTTP forme le cœur de nombreuses applications importantes dans le monde entier, et probablement personne ne peut dire qu'il en maîtrise tous les aspects.

Si vous travaillez sur Internet, vous utilisez très certainement Apache HTTP occasionnellement, et plus probablement tout le temps. Des mainteneurs en assurent le support, et les problèmes sont gérables mais fréquents, comme ce doit être le cas pour tout projet aussi complexe.

Même les logiciels que vous achetez pour assurer la sécurité sont imparfaits et doivent être renforcés par d'autres mesures, une pratique que nous appelons la défense en profondeur, ou Defense-in-Depth (DiD). Fin 2023, le service d'authentification Okta a constaté que des attaquants avaient compromis ses systèmes, volé les identifiants d'un grand nombre d'utilisateurs et étaient parvenus, comme l'écrivait KrebsOnSecurity, « à voler les jetons d'authentification de certains clients d'Okta, que les attaquants ont ensuite pu utiliser pour apporter des modifications aux comptes des clients, comme l'ajout ou la modification d'utilisateurs autorisés. »

Figure 2 : Approche et visibilité fondées sur les politiques d'entreprise



Dans le cadre de son analyse de l'incident, Okta a annoncé une meilleure prise en charge de certaines fonctionnalités considérées comme de bonnes pratiques dès lors qu'elles sont disponibles :

- Authentification multifacteur pour tous les comptes.
- Liaison IP – dans la mesure du possible, n'autorisez que les connexions en provenance ou à destination de services utilisant des adresses IP considérées comme légitimes. Ceci est une façon d'empêcher la réutilisation de jetons HTTP détournés, une autre possibilité dont disposaient les attaquants.
- En règle générale, il convient d'éliminer les privilèges permanents. Il s'agit de privilèges par défaut, mais souvent inutiles pour un compte. Réduisez les privilèges du compte au strict nécessaire pour effectuer les tâches qui lui sont associées.

N° 2 – Garantir l'authenticité et l'inaltérabilité de votre logiciel

La signature de code est une mesure de sécurité élémentaire qui fait ses preuves depuis plus de 30 ans. Elle utilise la cryptographie pour créer une signature numérique d'un logiciel (exécutable, bibliothèque, application, etc.). Les utilisateurs qui reçoivent le package peuvent utiliser un logiciel de cryptographie et la clé publique du distributeur pour prouver que le logiciel est authentique, c'est-à-dire que le prétendu éditeur du logiciel l'a effectivement créé et que le code n'a pas été modifié par un tiers. De nombreux systèmes d'exploitation, comme iOS, macOS et Windows, vérifient la signature numérique des logiciels avant leur installation et leur exécution.

La signature de logiciels est si 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.

Les clés de signature de code volées ont ainsi été un élément essentiel du ver Stuxnet, une attaque d'un acteur étatique qui a saboté les installations iraniennes d'enrichissement d'uranium. Révélée publiquement en 2010, Stuxnet était une attaque sophistiquée utilisant quatre vulnérabilités « zero-day » de Windows.

Comme pour toutes les applications PKI, il est essentiel que le signataire protège les clés de signature du code privé. C'est pourquoi le Certificate Authority/Browser Forum exige que les clés privées associées aux certificats de signature de code publiquement approuvés soient stockées dans un dispositif FIPS 140 de niveau 2, Common Criteria EAL 4+, tel qu'un module de sécurité matériel (HSM).

Si la sécurisation du stockage des clés est une première étape nécessaire, la sécurisation de la signature de code au sein d'une entreprise requiert des précautions supplémentaires, ne serait-ce que parce que la simple compromission d'un nom d'utilisateur/mot de passe pourrait permettre à un acteur malveillant 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 et clés de signature de code, y compris de leur rotation et de leur révocation
- Processus d'approbation qui détermine non seulement les utilisateurs autorisés à utiliser 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 effectuer des opérations de signature ; 2) les utilisateurs autorisés à approuver les opérations de signature ; et 3) les utilisateurs autorisés à superviser le processus
- Lorsque des approbations sont nécessaires, le système doit permettre la mise en place d'approbations multiples, selon un système « m sur n », c'est-à-dire une approbation par un certain nombre (m) de personnes sur un ensemble de n utilisateurs autorisés à approuver.
- 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, généralement les bonnes pratiques décrites dans ce document, à appliquer aux logiciels avant leur déploiement
- Signature des artefacts logiciels intermédiaires tout au long du cycle de développement, incluant le code source avant validation, les scripts et scripts de test, des conteneurs et des autres fichiers liés à l'infrastructure logicielle
- Plateforme centralisée pour gagner en 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

Si vous avez besoin d'une politique de sécurité relative à la signature de code, nous pouvons vous aider. Commencez par télécharger notre modèle afin de l'adapter aux spécificités de votre organisation.

N° 3 – Recherche de menaces, de vulnérabilités et de malwares

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 build 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.

Le code malveillant intégré à une bibliothèque open-source est un problème qui existe depuis longtemps et qui est à l'origine de nombreuses vulnérabilités dans la supply chain logicielle. Dans son rapport sur L'état de la sécurité de la supply chain logicielle, reversingLabs décrit une campagne appelée « Operation Brainleeches » qui a débuté en mai 2023. Brainleeches a utilisé des packages npm pour héberger des fichiers utilisés dans de vastes campagnes de phishing ciblant les clients Microsoft 365 dans le but d'obtenir leurs identifiants de connexion. Il existe d'innombrables exemples de tels malwares dans des dépôts publics et de confiance de code open-source.

Divers outils de sécurité déjà disponibles aident à résoudre certains de ces problèmes, tels que les outils SAST (tests statiques de la sécurité applicative). Néanmoins, ces outils tendent à se concentrer sur un seul aspect (tel que les vulnérabilités dans les logiciels open-source) ou une étape particulière du cycle de développement logiciel (SDLC).

Ce dont les entreprises ont besoin, c'est d'une méthode complète d'analyse du code source et 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. De nombreuses entreprises innovent dans le domaine de la prévention anti-malware, par exemple en proposant des dépôts de code source nettoyés.

N° 4 – Assurer la transparence du logiciel avec la SBOM

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 référencent tous les composants d'un package logiciel donné, quelle que soit son origine (open-source, bibliothèque commerciale, etc.). Les SBOM ne se contentent pas d'énumérer une liste « de composants logiciels ». Elles servent également à évaluer la qualité de ces composants, telles que les versions utilisées, les vulnérabilités connues, etc.

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

- **Comprendre les changements d'une version à une autre :** Les professionnels de la tech ne connaissent que trop la frustration que génère la résolution de problèmes entre deux versions de logiciels prétendument identiques. Tout comme son équivalent physique dans l'industrie, une nomenclature logicielle permet aux organisations de comprendre quels éléments sont nécessaires pour reproduire leur logiciel et comment ces composants évoluent dans le temps.
- **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

Ne vous contentez pas de générer des SBOM, demandez à vos fournisseurs de logiciels de vous les fournir et implémentez un mécanisme de vérification dans le cadre de votre évaluation. Certaines SBOM peuvent être signées numériquement dans leur intégralité ou par composant.

Lorsque vous déployez des SBOM, vous contrecarrez activement les plans des cybercriminels qui tentent d'exploiter le flou ambiant pour insérer un code nuisible dans vos logiciels. Une nomenclature logicielle est l'une des méthodes les plus efficaces pour protéger votre entreprise.

N° 5 – Automatiser grâce à plus de visibilité et plus de contrôle

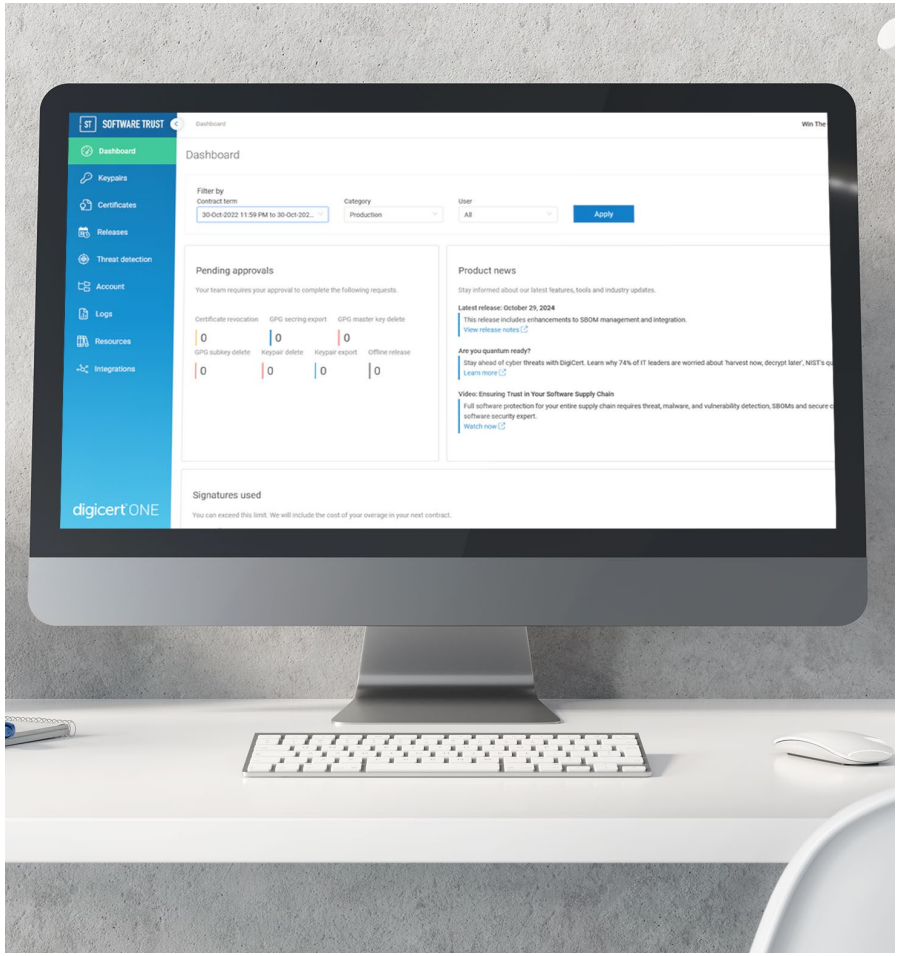
Vous avez peut-être remarqué que de nombreuses opérations de sécurité doivent être réalisées de manière récurrente. Les experts en sécurité s'accordent à dire que la seule façon d'y parvenir est d'automatiser les processus.

Dans le cycle de développement logiciel, les opérations de sécurité doivent être intégrées au processus, généralement par le biais d'un pipeline CI/CD. Ces systèmes sont conçus pour inclure des opérations de sécurité et pour mettre en place des workflows en fonction des résultats attendus.

La bonne approche consiste à faire en sorte que ces processus soient pilotés en accord avec les politiques d'entreprise et intégrés aux autres systèmes de sécurité de l'entreprise. Grâce à une visibilité et un contrôle centralisés, les entreprises peuvent renforcer leur conformité et leurs capacités de reporting, tout en améliorant la sécurité globale. En outre, il est essentiel d'implémenter ces processus de manière à faciliter et rationaliser les workflows des développeurs, plutôt que de les alourdir, afin de booster l'efficacité et la sécurité de l'environnement de développement.

Être agile ou faillir

La sécurité de la supply chain logicielle est un éternel recommencement. Les possibilités d'attaque sont nombreuses et vous devez faire confiance à la sécurité mise en place par les autres parties. Les bonnes pratiques exigent que la sécurité soit intégrée à chaque étape dans le processus CI/CD, à l'aide d'une grande variété de techniques de sécurité. La seule façon d'y parvenir est d'automatiser les processus. Un haut niveau d'automatisation permettra non seulement de prévenir les incidents en garantissant l'application régulière des mesures de sécurité, mais aussi de réagir aux incidents, de modifier les processus si nécessaire et de générer les informations nécessaires à l'établissement des rapports de conformité.



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 gestion des versions et de déploiement. 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 cinq bonnes pratiques en toute simplicité.

Découvrez comment protéger vos logiciels avec Software Trust Manager

©2025 DigiCert, Inc. Tous droits réservés. DigiCert est une marque déposée de DigiCert, Inc. aux États-Unis et dans d'autres pays. Toutes les autres marques commerciales et marques déposées sont la propriété de leurs propriétaires respectifs.