Move2Cloud

WHITE PAPER — ÉDITION 2027

DEVSECOPS & SÉCURITÉ APPLICATIVE

Supply Chain Logicielle
Le Nouveau Front de la Cybersécurité

SBOM, sécurisation du pipeline CI/CD, signature du code et gestion continue des dépendances open source — comment structurer une démarche DevSecOps face à une menace en forte croissance et une échéance réglementaire européenne.

11 pages · Comparatif des contrôles CI/CD, grille d'auto-évaluation et feuille de route incluses

30%
des compromissions impliquent un tiers en 2025, contre 15 % un an plus tôt (Verizon DBIR)

4,91 M$
coût moyen d'une compromission de la chaîne d'approvisionnement logicielle

+75%
de nouveaux paquets open source malveillants détectés en un an (Sonatype)

PRÉPARÉ PAR
Move2Cloud — Conseil Cloud & Infrastructure, Paris

PUBLICATION
Septembre 2026

SOMMAIRE

Ce que vous allez trouver dans ce document

Un parcours en six temps, de la compréhension de la menace à une feuille de route DevSecOps opérationnelle.

À qui s'adresse ce document ? DSI, RSSI, équipes plateforme et de développement dont les applications dépendent — comme la quasi-totalité des applications modernes — de composants open source et d'un pipeline CI/CD.

SYNTHÈSE

L'essentiel en une page

La sécurité applicative ne se joue plus seulement dans le code que l'on écrit — elle se joue tout autant dans le code que l'on assemble, souvent sans le savoir précisément.

Selon le rapport 2025 Data Breach Investigations de Verizon, **30 % des compromissions impliquaient un tiers** — un doublement par rapport à 15 % l'année précédente, décrit comme le plus fort basculement en un an de l'histoire du rapport. Une compromission de la chaîne d'approvisionnement logicielle coûte en moyenne **4,91 millions de dollars**, environ 11 % de plus que la moyenne générale des incidents, et met en moyenne **267 jours** à être identifiée et contenue — le délai de détection le plus long de tous les vecteurs d'attaque suivis.

Cette pression vient aussi du volume : selon Sonatype, plus de **454 600 nouveaux paquets open source malveillants** ont été identifiés en 2025 sur les registres npm, PyPI, Maven, NuGet et Hugging Face — une hausse de 75 % sur un an, avec plus de 99 % de ce volume concentré sur le seul registre npm. Face à cela, le cadre réglementaire européen se durcit : le Cyber Resilience Act impose déjà, depuis septembre 2026, une obligation de notification des vulnérabilités, avec une conformité complète — SBOM, documentation technique, marquage CE — exigée pour décembre 2027.

30%
des compromissions impliquent un tiers en 2025, contre 15 % un an plus tôt

4,91 M$
coût moyen d'une compromission de la chaîne d'approvisionnement logicielle

267 j.
délai moyen de détection — le plus long de tous les vecteurs d'attaque

Pour toute organisation qui publie ou exploite du logiciel — ce qui, avec des applications composées en moyenne de plus de 80 % de code open source, concerne pratiquement toutes les organisations — la question n'est plus « utilisons-nous des dépendances tierces ? » mais « savons-nous précisément lesquelles, et comment elles sont sécurisées tout au long du pipeline ? ». Ce document propose un cadre pour y répondre.

SECTION 02

Pourquoi la chaîne d'approvisionnement logicielle est devenue la cible n°1

Une application moderne n'est plus, pour l'essentiel, du code écrit en interne : elle est un assemblage de dépendances open source, de conteneurs de base, d'outils CI/CD et de services tiers. Cette composition, qui a démultiplié la vitesse de développement, a aussi démultiplié la surface d'attaque — et déplacé la cible des attaquants.

1. Compromettre un composant, c'est compromettre tous ses utilisateurs

Un attaquant qui parvient à injecter du code malveillant dans un paquet open source largement utilisé n'a plus besoin de cibler chaque organisation individuellement : la compromission se propage automatiquement à chaque build qui intègre ce paquet. C'est cette mécanique de propagation qui rend l'attaque de chaîne d'approvisionnement disproportionnellement efficace par rapport à son coût pour l'attaquant.

2. La détection reste structurellement tardive

Un délai moyen de 267 jours pour identifier une compromission de la chaîne d'approvisionnement signifie, concrètement, que le code malveillant a eu le temps d'être déployé en production, potentiellement à plusieurs reprises, avant sa découverte. Cette latence tient à la difficulté d'auditer manuellement des milliers de dépendances transitives.

3. Les registres open source sont eux-mêmes devenus des cibles directes

La croissance de 75 % des paquets malveillants détectés en un an ne reflète pas seulement plus d'attaques opportunistes : elle reflète une industrialisation de la publication de paquets typosquattés ou compromis, avec une concentration très marquée sur les registres les plus utilisés comme npm.

« La question n'est plus de savoir si un composant de votre chaîne d'approvisionnement sera un jour ciblé — c'est de savoir si vous serez capable de le détecter avant qu'il n'atteigne la production. »

SECTION 03

Le SBOM : de bonne pratique à obligation réglementaire

Un SBOM (Software Bill of Materials) est l'inventaire structuré et lisible par machine de tous les composants qui entrent dans la composition d'un logiciel — la condition préalable à toute gestion sérieuse du risque de chaîne d'approvisionnement.

Sans SBOM à jour, une organisation découvre généralement qu'elle est exposée à une vulnérabilité critique de la même façon que le grand public : par une alerte médiatisée, suivie d'une recherche manuelle et urgente pour savoir si le composant concerné est utilisé quelque part dans son parc applicatif. Avec un SBOM maintenu en continu, cette même question se résout par une simple requête.

ÉCHÉANCE	CE QU'ELLE IMPOSE
11 septembre 2026	Entrée en vigueur des obligations de notification des vulnérabilités et incidents pour les fabricants de produits comportant des éléments numériques, au titre du Cyber Resilience Act (CRA) européen.
11 décembre 2027	Conformité complète exigée : SBOM machine-lisible couvrant au minimum les dépendances de premier niveau, documentation technique, évaluation des risques et marquage CE.

> **Point de vigilance :** le CRA s'applique à toute entreprise qui met un produit numérique sur le marché européen, quel que soit le lieu de son siège social — les éditeurs français ne peuvent pas se considérer hors périmètre du seul fait de leur implantation.

Au-delà de la conformité : un outil de réponse à incident

Le SBOM n'est pas qu'une obligation documentaire : lorsqu'une vulnérabilité critique est révélée dans une bibliothèque largement utilisée, disposer d'un SBOM à jour permet de répondre en heures — plutôt qu'en semaines — à la question qui compte vraiment : sommes-nous exposés, et où ?

SECTION 04

Shift-left : intégrer la sécurité dans le pipeline CI/CD

Le principe du « shift-left » consiste à déplacer les contrôles de sécurité le plus tôt possible dans le cycle de développement — là où une correction coûte des minutes, pas des jours de production perdue.

ÉTAPE DU PIPELINE	CONTRÔLE À INTÉGRER
Commit / pull request	Analyse statique du code (SAST) et détection de secrets codés en dur, exécutées automatiquement avant toute fusion vers la branche principale.
Build	Génération automatique du SBOM et analyse de composition logicielle (SCA) pour détecter les dépendances vulnérables ou obsolètes.
Conteneurisation	Scan des images de conteneurs avant publication dans un registre, avec blocage automatique des images contenant des vulnérabilités critiques connues.
Déploiement	Vérification de la signature et de la provenance des artefacts avant tout déploiement en production (détaillé en section 05).

> **À retenir :** l'objectif du shift-left n'est pas d'ajouter des contrôles supplémentaires en fin de cycle, mais de faire échouer un build dès qu'un problème est détecté — le coût de correction d'une vulnérabilité augmente fortement à chaque étape franchie sans être détectée.

Éviter l'écueil du tout-bloquant

Un pipeline qui bloque systématiquement sur chaque alerte, y compris mineure, pousse rapidement les équipes à contourner les contrôles plutôt qu'à les respecter. Une politique de sévérité graduée — blocage automatique sur le critique, alerte non bloquante sur le mineur — maintient l'adhésion des équipes de développement dans la durée.

SECTION 05

Signature et provenance : garantir l'intégrité du code déployé

Savoir qu'un composant est sain au moment de son intégration ne suffit pas — encore faut-il garantir que l'artefact réellement déployé en production est celui qui a été validé, sans altération intermédiaire.

Signer chaque artefact produit par le pipeline

Chaque image de conteneur, chaque paquet et chaque binaire produit par le pipeline CI/CD devrait être signé cryptographiquement au moment de sa construction, avec une vérification systématique de cette signature avant tout déploiement. Cette pratique empêche qu'un artefact modifié après coup — que ce soit par une compromission du registre ou de l'infrastructure de déploiement — soit exécuté sans détection.

Tracer la provenance, pas seulement l'intégrité

Au-delà de la signature, une attestation de provenance documente précisément comment un artefact a été construit : à partir de quel code source, avec quels outils, sur quelle infrastructure. Cette traçabilité permet de détecter une construction anormale — un artefact produit en dehors du pipeline officiel, par exemple — avant même d'en analyser le contenu.

CONTRÔLE	CE QU'IL GARANTIT
Signature cryptographique	L'artefact déployé est identique à celui produit et validé par le pipeline, sans altération intermédiaire.
Attestation de provenance	L'artefact provient bien du pipeline officiel, à partir du code source attendu, et non d'une source parallèle ou compromise.
Vérification au déploiement	Aucun artefact non signé ou à la provenance non vérifiée ne peut atteindre un environnement de production.

SECTION 06

Gérer les dépendances open source en continu

L'inventaire ponctuel des dépendances se périme dès sa publication — la gestion des dépendances open source n'a de valeur que si elle est continue.

Mettre à jour, pas seulement détecter

Détecter une dépendance vulnérable sans processus établi pour la mettre à jour ne fait que déplacer le problème vers un tableau de bord ignoré. L'automatisation des mises à jour de dépendances mineures et de correctifs de sécurité — avec des tests automatisés validant l'absence de régression — reste le moyen le plus fiable de maintenir un parc applicatif à jour sans mobiliser une équipe dédiée en continu.

Évaluer la santé du projet, pas seulement sa popularité

Le nombre de téléchargements d'un paquet n'indique rien sur sa maintenance réelle. Des signaux plus fiables : la fréquence des correctifs de sécurité publiés, le nombre de mainteneurs actifs, et l'existence d'un historique de réponse rapide aux vulnérabilités signalées — des critères à intégrer dès le choix initial d'une dépendance, pas seulement lors d'un audit a posteriori.

Limiter le rayon d'action d'une compromission

Au-delà de la prévention, limiter les permissions accordées aux processus de build et de déploiement — accès réseau restreint, jetons d'authentification à durée de vie courte et scope minimal — réduit l'impact d'une dépendance malveillante qui échapperait aux contrôles précédents, sans empêcher totalement l'incident.

> **À retenir :** aucun contrôle pris isolément — scan, signature, mise à jour automatisée — ne suffit à éliminer le risque de chaîne d'approvisionnement. C'est leur combinaison, appliquée en continu plutôt qu'en audit ponctuel, qui réduit réellement l'exposition.

SECTION 07

Grille d'auto-évaluation DevSecOps

Avant de considérer votre chaîne d'approvisionnement logicielle comme maîtrisée, vérifiez ces sept contrôles.

- [ ] **Un SBOM à jour existe-t-il pour chacune de vos applications critiques ?** Sans lui, répondre à « sommes-nous exposés ? » lors d'une alerte de sécurité prend des jours plutôt que des heures.
- [ ] **L'analyse de composition logicielle (SCA) est-elle exécutée automatiquement à chaque build ?** Une analyse ponctuelle ne détecte pas une dépendance devenue vulnérable après son intégration.
- [ ] **Chaque artefact déployé en production est-il signé et sa signature vérifiée avant exécution ?** Sans cela, rien ne garantit que l'artefact déployé est celui qui a été validé.
- [ ] **Les mises à jour de sécurité des dépendances sont-elles automatisées, ou dépendent-elles d'une revue manuelle périodique ?** Une revue manuelle prend systématiquement du retard face au volume de paquets à jour.
- [ ] **Les jetons d'accès utilisés par le pipeline CI/CD ont-ils une portée et une durée de vie limitées ?** Un jeton trop permissif transforme une compromission mineure du pipeline en accès étendu.
- [ ] **Votre organisation sait-elle si elle entre dans le périmètre du Cyber Resilience Act ?** Tout éditeur commercialisant un produit numérique dans l'UE peut être concerné, indépendamment de sa taille.
- [ ] **Existe-t-il une procédure documentée de réponse en cas de compromission détectée d'une dépendance ?** Sans procédure préétablie, la réponse à incident improvise sous pression, au moment où l'erreur coûte le plus cher.

> **Comment lire vos réponses :** trois « non » ou plus indiquent qu'un audit structuré de la chaîne d'approvisionnement logicielle devrait précéder toute extension supplémentaire du parc applicatif exposé.

SECTION 08

Feuille de route en 5 étapes

Une démarche progressive, pensée pour intégrer la sécurité de la chaîne d'approvisionnement sans ralentir le rythme de livraison des équipes.

1

Inventorier

Générer un premier SBOM pour chaque application critique, afin d'objectiver précisément la composition réelle du parc applicatif.

2

Automatiser la détection

Intégrer l'analyse de composition logicielle (SCA) et le scan de conteneurs directement dans le pipeline CI/CD, avec une politique de sévérité graduée.

3

Signer et vérifier

Mettre en place la signature systématique des artefacts et le blocage au déploiement de tout artefact non vérifié.

4

Automatiser les mises à jour

Déployer un outil de mise à jour automatisée des dépendances, couplé à des tests automatisés garantissant l'absence de régression.

5

Préparer la conformité CRA

Formaliser la génération et la maintenance continue du SBOM, et documenter le processus de gestion des vulnérabilités en vue de l'échéance de décembre 2027.

CONCLUSION

La sécurité de la chaîne d'approvisionnement, un chantier continu

Avec 30 % des compromissions impliquant désormais un tiers et un cadre réglementaire européen qui se durcit, la sécurité de la chaîne d'approvisionnement logicielle n'est plus un sujet réservé aux grandes organisations disposant d'équipes de sécurité dédiées.

Les organisations qui intègrent SBOM, scan continu, signature et mise à jour automatisée directement dans leur pipeline — plutôt qu'en audit ponctuel — réduisent à la fois leur exposition réelle et le délai de réponse le jour où un incident survient malgré tout.

Move2Cloud	est un cabinet de conseil spécialisé en infrastructure, cloud et transformation IT. Nous accompagnons les organisations dans l'audit, la sécurisation et la mise en conformité de leur chaîne d'approvisionnement logicielle.
Contact	Move2Cloud — Paris · move2cloud.fr
Prochaine étape	Un audit DevSecOps permet de répondre en quelques semaines aux sept questions de la page 9, avec un plan de remédiation priorisé.

Sources

Verizon, « 2025 Data Breach Investigations Report » (synthèse via swif.ai, « Supply Chain Attack Statistics for 2026 ») — Sonatype, « 2026 State of the Software Supply Chain Report » — Mend.io, « EU Cyber Resilience Act: 2026 Compliance Guide » — règlement (UE) 2024/2847 (Cyber Resilience Act).