

Trois semaines pour lancer AppSec, le plan le plus simple pour améliorer votre profil de sécurité

05 janvier 2021, 01:00

Nick Boucart
Tatiana Galibus

« Shift security left » est un paradigme répandu dans le secteur informatique. S'il est très facile à comprendre, il n'est pas évident à mettre en œuvre. Son adoption exige plus que la simple utilisation des technologies : il s'agit d'un changement de culture, d'une approche intégrée de la sécurité des applications et d'un processus d'apprentissage continu. La plupart des start-up sont désireuses de l'adopter, mais elles tombent sur des obstacles lors de son application dans la pratique.

Les stratégies existantes sont peut-être adaptées aux entreprises ayant un profil de sécurité défini, mais il n'existe pas de lignes directrices communes sur la manière d'aborder la sécurité des applications, surtout si les ressources et l'expertise de la start-up sont limitées. Quelles sont les mesures de base à prendre pour disposer d'un plan d'amélioration de la maturité de la sécurité clair et être prêt à répondre aux questions des clients sur la confiance et la sécurité ?

Animés par la passion de trouver une solution pragmatique qui fonctionne vraiment pour les start-ups, nous avons mené un travail de recherche et de réflexion de près de trois mois chez Sirris. Résultat : *le plan le plus simple d'amélioration du profil de sécurité en 3 semaines*. Il s'inspire de la philosophie *DevSecOps* et des normes *OWASP*.



Nous avons expliqué les actions menées durant la première semaine dans le précédent article. À présent, que faut-il faire durant la deuxième semaine du plan le plus simple d'amélioration de la sécurité ?

Deuxième semaine : définir les barrières qualité et intégrer les outils d'analyse statique de code à votre chaîne de compilation

La deuxième semaine de pratique se consacre à vos premiers pas vers *l'automatisation de la sécurité*. Il existe des centaines d'outils open source et commerciaux pour l'analyse de sécurité des applications/sites Web. Parmi eux, les outils qui analysent le code et ses dépendances, et non l'application en cours d'exécution, sont appelés outils **SAST pour le test statique de sécurité des applications (Static Application Security Testing)**. Ces derniers se divisent en plusieurs catégories :

1. Scanners de codes

Les scanners vérifient votre code selon les règles de codage intégrées et vous informent de tout problème de qualité du code ou de toute faille de sécurité. Ces outils peuvent disposer de milliers de règles pour chaque langage de programmation et différents algorithmes de balayage. Certains d'entre eux sont orientés sécurité : ils fournissent une analyse approfondie des failles de sécurité. D'autres, plus axés sur la qualité de code, vous notifient d'éventuels problèmes ou inexactitudes en matière de codage. Nous vous conseillons de commencer avec l'un des outils suivants :

SonarScanner	Scanne les problèmes de sécurité et de qualité d Disponible en open source comme en version commerciale, il peut s'intégrer comme plug-in da environnement de développement pour faciliter s utilisation.

Fortify	Outil open source orienté sécurité. S'intègre facil à l'environnement de développement et au pipeli d'intégration continue et de déploiement continu (CI/CD).

2. Outils d'analyse de composition logicielle (SCA - Software Composition Analysis)

Les outils SCA scannent les bugs et les failles de sécurité dans les dépendances de codes et les composants tiers. Même si votre code répond aux exigences de sécurité, un appel à fonctions non sécurisées comme Java Random.NextInt() peut constituer une source de vulnérabilité. Nous avons souvent tendance à occulter l'utilisation d'API et de bibliothèques logicielles sécurisées. Nous vous recommandons d'utiliser les outils de balayage de dépendances suivants :

OWASP DependencyCheck	Scanne les dépendances non sécurisées ; intégré à SonarQube.
Fortify	Scanne les dépendances et bibliothèques logicielles non sécurisées ; plug-in de balayage de codes de Fortify.
Retire.js	Recherche l'utilisation de bibliothèques logicielles et de composants non sécurisés dans Java.
Bundler-audit	Recherche les versions vulnérables des composants de Ruby.
Gemnasium/GitLab security scan	Surveille les versions des composants dans différents langages de programmation et vous notifie de tout changement. À l'heure actuelle, cet outil s'intègre au pipeline d'automatisation CI/CD de GitLab.

3. Outils de détection des secrets

Il arrive souvent que les développeurs affectent involontairement des secrets, jetons d'API et identifiants aux répertoires distants. Les informations sensibles s'exposent alors facilement et peuvent être utilisées par des adversaires pour voler l'identité et obtenir un accès non autorisé à des ressources spécifiques. Les outils de détection des secrets scannent le code et détectent la présence de clés, de jetons et d'identifiants. Nous recommandons d'utiliser les outils suivants (la plupart étant intégrés à l'environnement CI/CD correspondant) :

GitGuardian	Automatiquement intégré à l'environnement de répertoire de GitLab.

GittyLeaks	Automatiquement intégré à l'environnement de répertoire de GitHub repository.
Git-secrets	Empêche les développeurs d'affecter des secrets répertoire de Git.
Truffle Hog	Cet outil scanne les informations sensibles du répertoire, y compris dans toutes ses branches.

4. Tests de fumée et outils d'analyse de conteneurs

Les tests de fumée ou les outils de vérification de versions (BVT) révèlent des erreurs simples assez graves pour rejeter la sortie d'un logiciel. Les outils de balayage de conteneurs Docker s'avèrent essentiels pour repérer les bibliothèques logicielles potentiellement vulnérables de votre conteneur. Ces outils détectent les insécurités et les bugs dans les images Docker avant le déploiement du code. Nous vous recommandons d'utiliser les outils suivants pour les tests de fumée et le balayage de conteneurs :

Clair-scanner	Outil de balayage de conteneurs intégré au pipel CI/CD de GitLab.
Trivy	Scanne les vulnérabilités de conteneurs, en déte les problèmes avec les paquets SE et les dépend applicatives
Twistlock	Assure une sécurité des conteneurs en cycle cor intégré à un système de gestion des vulnérabilité sur le cloud

5. Gestionnaires de vulnérabilités

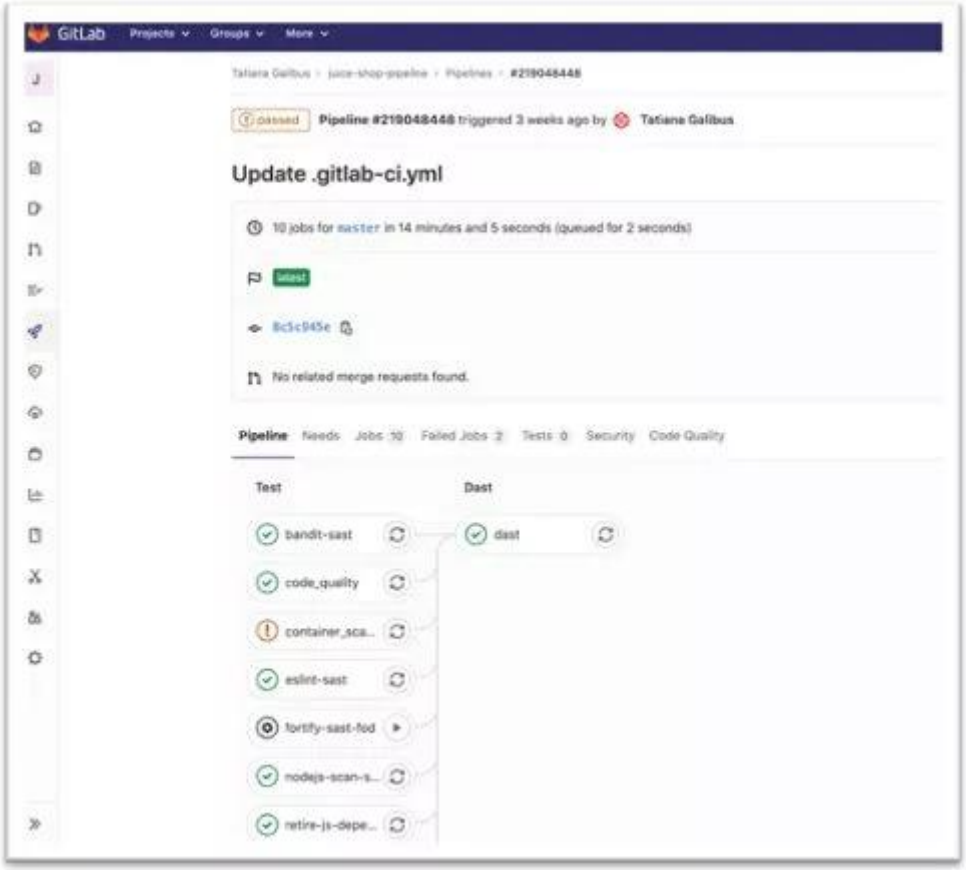
La gestion des vulnérabilités tient une place indispensable dans l'intégration continue dans la mesure où elle simplifie la surveillance continue et l'identification du code, des dépendances, des conteneurs, des images et des hôtes dans leur environnement, ainsi que la prévention des risques en leur sein. Les outils SAST s'intègrent souvent aux gestionnaires de vulnérabilités, ce qui permet d'avoir une idée approfondie des problèmes de sécurité. Nous vous recommandons les gestionnaires de vulnérabilités suivants :

SonarQube server	Affiche tous les problèmes détectés, classés par de sévérité et par type dans un tableau de bord p Permet de suivre, visualiser et analyser les vulnérabilités.

DefectDojo	Une plateforme de gestion de vulnérabilités qui p d'intégrer et d'orchestrer les résultats de plusieur scanners de sécurité
Tenable.io	Outil de gestion de vulnérabilités intégré à GitLab Fournit un tableau de bord automatique et un rap bugs de sécurité.

La force des outils SAST réside, entre autres, dans leur large couverture de langages de programmation et de plateformes de développement. Ils s'adoptent et s'intègrent assez facilement. En dépit d'un grand nombre de faux positifs, les outils SAST identifient près de 50 % de bugs et se montrent particulièrement efficaces dans l'amélioration de la qualité du code et dans la détection de problèmes de sécurité récurrents.

Ces outils constituent la base des pipelines de sécurité applicative. Le pipeline AppSec correspond à une chaîne d'outils de sécurité essentiels, intégrés à toutes les étapes du cycle de vie du



sécurité continue aux étapes

Exemple de pipeline GitLab, d'outils SAST et d'étape de test

Envie d'en savoir plus à ce sujet et sur les autres étapes ? D'autres questions pratiques sur la sécurité applicative ? Contactez-nous à l'adresse security@sirris.be.



]]>

Auteurs



Nick Boucart



Tatiana Galibus