DevSecOps

TP 2 - La sécurité

Pierre-Emmanuel De Clercq et Théo Saelens

Veille de la Sécurité dans le Cycle DevOps

Introduction

La méthodologie DevSecOps, fusion de développement (Dev), sécurité (Sec) et opérations (Ops), incarne l'intégration de la sécurité à chaque étape du cycle de développement logiciel. Cette approche ne se contente pas d'automatiser et d'intégrer les pratiques de sécurité dans le processus DevOps; elle vise aussi à créer une culture où la sécurité est une responsabilité partagée, permettant de détecter et corriger rapidement les vulnérabilités. Ainsi, elle améliore la qualité du code et réduit le risque de failles de sécurité dans les applications livrées.

I. Vérification de la sécurité des dépendances

Objectif

L'objectif est d'identifier et de corriger les vulnérabilités au sein des bibliothèques et packages utilisés, minimisant ainsi le risque d'exploitations.

Outils recommandés

Snyk: Cet outil analyse les dépendances des projets pour détecter les vulnérabilités connues, offrant des recommandations pour leur correction.

OWASP Dependency-Check: Outil open source qui scanne les projets pour identifier les dépendances vulnérables, utilisant les données du National Vulnerability Database (NVD).

Fonctionnement

Ces outils examinent les fichiers de dépendances du projet pour lister les bibliothèques utilisées, vérifiant si elles contiennent des vulnérabilités connues. Des rapports détaillés sont fournis avec des recommandations pour la mise à jour ou le remplacement des dépendances problématiques.

II. Vérification de la qualité du code

Objectif

Garantir que le code source adhère aux bonnes pratiques de programmation, identifiant les erreurs, les "code smells" et les problèmes de performance potentiels.

Outils recommandés

SonarQube: Détecte les bugs, les vulnérabilités et les mauvaises pratiques dans le code.

Codacy: Service de revue automatisée qui identifie les problèmes de sécurité, de performance et de style.

Fonctionnement

Ces outils évaluent le code source pour détecter des patterns problématiques et vérifient le respect des standards de codage. Ils peuvent s'intégrer aux pipelines CI/CD pour une évaluation continue.

III. Sécurité des images Docker

Objectif

Assurer que les images Docker ne comportent pas de vulnérabilités qui pourraient être exploitées une fois déployées.

Outils recommandés

Docker Bench for Security: Vérifie les configurations des conteneurs contre les bonnes pratiques recommandées par Docker.

Clair: Analyse les images Docker à la recherche de vulnérabilités connues.

Fonctionnement

Ces outils analysent les images Docker pour détecter les vulnérabilités en se référant à des bases de données. Des rapports détaillés offrent des recommandations pour sécuriser les images.

IV. Vérification des mots de passe

Objectif

Veiller à l'application de politiques de mots de passe robustes pour prévenir les accès non autorisés.

Outils recommandés

Have I Been Pwned: Permet de vérifier si un mot de passe a été compromis.

zxcvbn: Évalue la force des mots de passe selon plusieurs critères.

Fonctionnement

Ces outils évaluent l'efficacité des politiques de mots de passe en place, offrant des métriques de sécurité et des recommandations pour renforcer les politiques.

V. Tests automatisés

Objectif

Automatiser l'exécution de tests pour identifier rapidement les problèmes de sécurité dans les applications.

Outils recommandés

Selenium: Automatise les tests sur les navigateurs web.

Jenkins: Serveur d'intégration continue configuré pour exécuter divers types de tests de sécurité de manière automatisée.

Fonctionnement

L'automatisation des tests utilise des scripts ou des frameworks pour effectuer des tests répétitifs, y compris unitaires, d'intégration, de performance et de vulnérabilité, s'intégrant aux pipelines CI/CD pour une correction rapide des problèmes de sécurité.

Conclusion

L'adoption de DevSecOps est cruciale pour le développement d'applications sécurisées. Les outils présentés fournissent une base solide pour intégrer la sécurité dans chaque phase du développement logiciel, favorisant ainsi la détection et la correction efficaces des vulnérabilités. Intégrer ces outils et pratiques de manière stratégique dans les pipelines de développement permet non seulement de sécuriser les applications mais aussi de promouvoir une culture de sécurité partagée au sein des équipes.