DevSecOps

PARTIE 2

CI/CD : Docker et Kubernetes

N° de la lecture individuelle :	2
Semestre	4
Étudiant	DAVID Guillaume, 803_1F
Sujet	DevSecOps – Partie 2 – CI/CD : Docker et Kubernetes

Support théorique

La recherche se base fondamentalement sur le livre présenté ci-dessous. Des apports, de l'aide à la construction des exemples, et des compréhensions ont également réalisés avec ChatGPT.

Titre

DevSecOps – Développez et administrez vos services en toute sécurité
Éditions ENI, 2023, ISBN : 9782409039188| 618 pages

Auteur

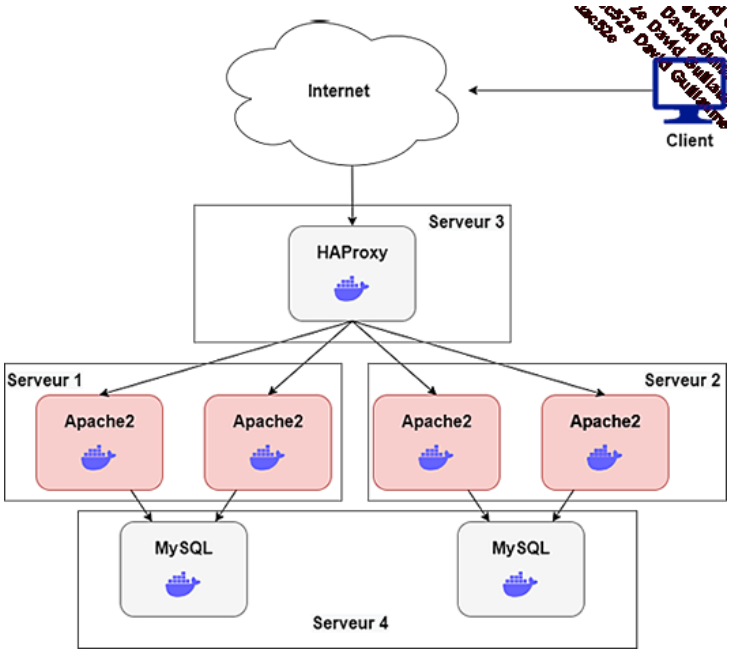
Jordan Assouline

Table des matières

Utilisation de Kubernetes en DevSecOps

Prise en main de Kubernetes

Le déploiement d'applications conteneurisées présente des défis de gestion complexes lorsqu'il s'agit de distribuer des services sur plusieurs serveurs, motivés par des besoins de répartition géographique, de performances accrues ou de haute disponibilité. Kubernetes, système open-source, répond à ces défis en automatisant le déploiement, la mise à l'échelle et la gestion des applications conteneurisées. En utilisant des pools de serveurs, Kubernetes simplifie le déploiement en permettant la configuration des ressources disponibles et en automatisant la réplication des services sur plusieurs serveurs pour garantir la haute disponibilité. De plus, Kubernetes offre des fonctionnalités avancées telles que la mise à l'échelle automatique en fonction de la demande, facilitant ainsi la gestion des conteneurs déployés à travers plusieurs serveurs, tout en offrant une vue d'ensemble et une simplification par rapport à une utilisation directe de Docker.

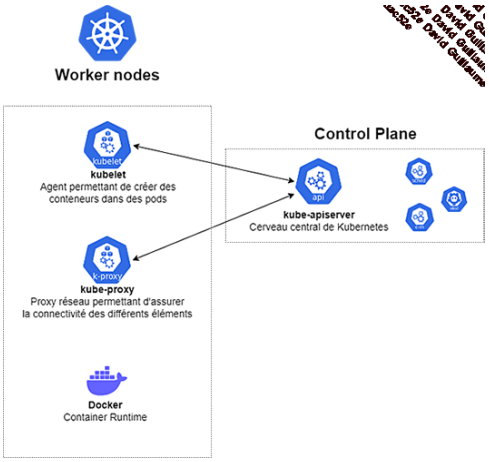


Le concept du cluster

Un cluster Kubernetes représente un ensemble de machines, appelées nœuds, sur lesquelles des conteneurs sont exécutés, facilitant ainsi le déploiement et la gestion d'applications

- Un cluster Kubernetes est un groupe de machines, appelées nœuds, sur lesquelles des conteneurs sont exécutés, simplifiant le déploiement et la gestion des applications conteneurisées.
- Les services essentiels tels que le kube-apiserver, l'etcd, le kube-scheduler et le kube-controller-manager sont regroupés dans ce qu'on appelle le Control Plane.
- Les Worker nodes hébergent les conteneurs et incluent des services comme le kubelet, le kube-proxy et le Container Runtime, avec Docker comme exemple courant.

- Cette structure facilite l'interaction continue entre les nœuds et le Control Plane, rendant ainsi la gestion et le déploiement des applications à travers le cluster Kubernetes plus efficaces.



Sécurité de Kubernetes

Pour garantir la sécurité du cluster Kubernetes, il est essentiel de vérifier sa configuration à l'aide des **CIS Benchmarks**, un ensemble de normes détaillées pour sécuriser Kubernetes, couvrant divers aspects tels que les composants du Control Plane, la configuration du Control Plane, les Worker nodes, les politiques de sécurité, etc.

Ces vérifications peuvent être fastidieuses, mais des outils tels que **Kube-Bench** peuvent automatiser ce processus en vérifiant la conformité du cluster aux recommandations du CIS Benchmark. Pour exécuter Kube-Bench, il faut télécharger et lancer un fichier de job YAML sur le cluster, puis vérifier les résultats à l'aide de la commande kubectl logs. Il est également important de corriger les erreurs identifiées, telles que les paramètres de configuration incorrects ou obsolètes.

```
Vulnerabilities
For further information about a vulnerability, search its ID in:
https://avd.aquasec.com/
+--------+--------------------+--------------------+--------------------+--------------------+----------+
| ID     | LOCATION           | MITRE CATEGORY     | VULNERABILITY      | DESCRIPTION        | EVIDENCE |
+--------+--------------------+--------------------+--------------------+--------------------+----------+
| KHV002 | 192.168.101.64:6443 | Initial Access //  | K8s Version        | The kubernetes     | v1.24.3  |
|        |                    | Exposed sensitive  | Disclosure         | version could be   |          |
|        |                    | interfaces         |                    | obtained from the  |          |
|        |                    |                    |                    | /version endpoint  |          |
+--------+--------------------+--------------------+--------------------+--------------------+----------+
| KHV002 | 10.132.0.2:6443    | Initial Access //  | K8s Version        | The kubernetes     | v1.24.3  |
|        |                    | Exposed sensitive  | Disclosure         | version could be   |          |
|        |                    | interfaces         |                    | obtained from the  |          |
|        |                    |                    |                    | /version endpoint  |          |
+--------+--------------------+--------------------+--------------------+--------------------+----------+
```

Figure 1: Sortie de kube-bench

Un autre outil utile pour identifier les vulnérabilités dans votre cluster est **Kube-hunter** qui permet de mettre en évidence les failles de sécurité potentielles en explorant les services exposés sur le cluster. Pour l'utiliser, il faut installer à l'aide de Python et PIP, puis exécuter en choisissant l'option d'analyse appropriée (par exemple, l'analyse d'interface pour scanner les sous-réseaux sur toutes les interfaces réseau locales). Une fois l'analyse terminée, un rapport détaillé sur les vulnérabilités détectées sera présenté.

Les objets dans Kubernetes

Dans Kubernetes, la configuration des éléments se fait principalement à travers des objets définis au format YAML. Les Pods, qui sont essentiels pour le déploiement et la gestion des conteneurs, sont créés en interagissant avec le kube-apiserver.

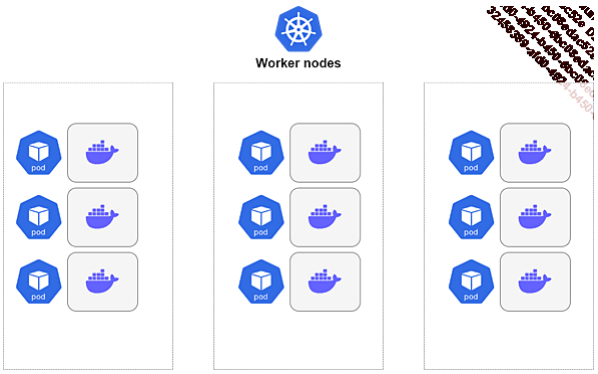
Les pods

Dans Kubernetes, les Pods sont utilisés pour encapsuler et gérer des conteneurs sur les Worker nodes. Bien que la plupart des Pods contiennent un seul conteneur, il est possible d'en avoir plusieurs dans un même Pod, tant qu'ils remplissent des fonctions distinctes.

Lorsqu'un Pod contient plusieurs conteneurs, ceux-ci partagent les mêmes ressources réseau et de stockage. Cette approche est utile dans divers scénarios, tels que

- **Applications composées de plusieurs services :** Par exemple, une application web avec un Front-End, un service de traitement d'images et une base de données. Regrouper ces services dans un même Pod permet une communication efficace entre eux sans avoir à transiter par plusieurs Pods distincts.
- **Utilisation de Sidecars :** Un conteneur, appelé Sidecar, peut être ajouté au Pod pour fournir une fonctionnalité supplémentaire, comme la gestion des logs ou l'accès à des ressources partagées. Par exemple, un conteneur Python pourrait être ajouté en tant que Sidecar pour accéder directement aux images téléchargées par l'application web dans le même Pod.
-

Cette approche simplifie la communication et la gestion des ressources entre les services, tout en maintenant une isolation raisonnable entre les différents composants de l'application.

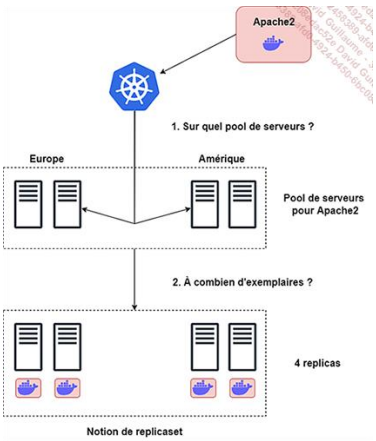


Les réplicas

Un ReplicaSet est un objet Kubernetes utilisé pour garantir qu'un nombre spécifié de répliques identiques (pods) d'une application s'exécutent simultanément. En d'autres termes, un ReplicaSet maintient un ensemble de pods identiques en fonctionnement, en veillant à ce qu'il y ait toujours un nombre défini de répliques disponibles, même en cas de panne ou d'arrêt involontaire.

Les ReplicaSets sont souvent utilisés pour garantir la disponibilité et la tolérance aux pannes des applications en s'assurant qu'un nombre prédéfini de répliques de l'application est toujours opérationnel, même en cas de défaillance d'un pod ou d'un noeud dans le cluster

Kubernetes. Si le nombre de répliques tombe en dessous du seuil spécifié, le ReplicaSet crée automatiquement de nouveaux pods pour maintenir le nombre désiré.



Les déploiements

Un déploiement (Deployment) dans Kubernetes est une ressource qui permet de déclarer l'état désiré d'une application dans un cluster Kubernetes. Il facilite la gestion du déploiement, de la mise à jour et du scaling des applications de manière déclarative. Concrètement, un déploiement spécifie le nombre de répliques de pods à exécuter ainsi que les spécifications des conteneurs, telles que l'image Docker à utiliser, les variables d'environnement, les volumes montés, etc. Kubernetes surveille l'état des pods en permanence et s'assure que le nombre spécifié de répliques est toujours en cours d'exécution, en redémarrant ou en recréant les pods si nécessaire pour atteindre l'état désiré.

Les services

Un service dans Kubernetes est une abstraction qui définit un ensemble de pods et une politique d'accès à ces pods. Il permet de fournir un point d'accès stable et unifié à un ensemble de pods, quelles que soient les instances individuelles qui les exécutent ou leur cycle de vie. Les services peuvent être de différents types, notamment

- **Service ClusterIP**
  Expose un ensemble de pods en interne au sein du cluster Kubernetes. Il fournit un IP virtuel stable et permet aux autres services dans le cluster de communiquer avec les pods via cet IP.
- **Service NodePort**
  Expose un service sur un port fixe sur chaque node du cluster. Il permet d'accéder aux services à partir de l'extérieur du cluster en utilisant l'adresse IP du node et le port spécifié.
- **Service LoadBalancer**
  Expose un service en utilisant un équilibreur de charge externe. Il permet d'accéder aux services à partir de l'extérieur du cluster via une adresse IP externe fournie par le fournisseur de services cloud.
- **Service ExternalName**
  Permet d'exposer un service en tant qu'enregistrement DNS externe. Il redirige le trafic vers l'adresse DNS spécifiée plutôt que vers des pods dans le cluster.

Les probes

Dans Kubernetes, les « probes » sont des mécanismes essentiels permettant de vérifier l'état de santé des conteneurs exécutés dans un pod. Ils sont utilisés pour déterminer si un conteneur est prêt à recevoir du trafic réseau ou s'il doit être redémarré en cas de dysfonctionnement. Il existe trois types de sondages : la Probe d'initialisation, la Probe de disponibilité et la Probe de fiabilité. La Probe d'initialisation vérifie si un conteneur est prêt à recevoir du trafic dès son démarrage, tandis que la Probe de disponibilité examine si le conteneur fonctionne correctement. En cas d'échec de la Probe de disponibilité, Kubernetes peut redémarrer le conteneur pour rétablir son état opérationnel. Enfin, la Probe de fiabilité vérifie si un conteneur peut gérer les requêtes entrantes, permettant ainsi à Kubernetes de diriger le trafic uniquement vers les conteneurs prêts à répondre.

Les sondages Kubernetes sont configurables via des options telles que les délais, les seuils de succès et d'échec, ainsi que les actions à prendre en cas de défaillance. Ces sondages sont cruciaux pour maintenir la disponibilité et la fiabilité des applications déployées dans un cluster Kubernetes, garantissant ainsi une meilleure gestion des conteneurs et une résilience accrue de l'infrastructure.

Les tests de lint sur les objets Kubernetes

KubeLinter est spécifiquement conçu pour Kubernetes. Il examine les fichiers YAML contenant les définitions d'objets Kubernetes et identifie les erreurs, les mauvaises pratiques et les problèmes de sécurité potentiels. Il vérifie la conformité aux meilleures pratiques, aux normes de sécurité et aux recommandations de Kubernetes, permettant ainsi aux développeurs et aux administrateurs de s'assurer que leur configuration est optimale et conforme aux normes établies.

Les tests de sécurité sur les objets Kubernetes

Checkov est principalement utilisé pour l'infrastructure as code (IaaC) et se concentre sur Terraform, bien qu'il prenne également en charge d'autres formats comme AWS CloudFormation. Checkov examine les fichiers de configuration IaC et détecte les problèmes de sécurité, les erreurs de configuration et les lacunes de conformité par rapport aux meilleures pratiques de sécurité. Il fournit des recommandations pour corriger les problèmes identifiés, aidant ainsi les équipes à maintenir un environnement cloud sécurisé et conforme.

Conclusion

En plus de KubeLinter et Checkov, d'autres outils comme Kubeaudit, Kube-scan et Kube-score sont disponibles pour effectuer des audits de sécurité, évaluer les risques et réaliser des analyses statiques de sécurité sur les objets Kubernetes. Tester ces outils sur vos clusters et vos objets vous permettra d'identifier et de corriger efficacement les vulnérabilités, renforçant ainsi la sécurité de votre infrastructure cloud.

Culture et connaissance en cybersécurité

Vocabulaire

Le CIA, pour Confidentialité, Intégrité et Disponibilité, représente les principes fondamentaux de la sécurité informatique. Confidentialité garantit que seules les personnes autorisées

accèdent aux données, Intégrité assure que les données ne sont pas altérées par des individus non autorisés, tandis que Disponibilité garantit que les données sont accessibles pour les utilisateurs autorisés quand ils en ont besoin. Ces principes servent de base pour minimiser les risques liés à la sécurité des réseaux et des infrastructures, et guident les politiques de sécurité pour assurer une "information assurance".

Le SIEM, ou Security Information and Event Management, combine la gestion des informations de sécurité et la gestion des événements de sécurité pour fournir une analyse en temps réel de la sécurité des systèmes. Il agrège et collecte des données provenant de différents équipements réseau, serveurs et applications, puis utilise un processus de corrélation pour détecter les comportements malveillants et les événements suspects. Le SIEM joue un rôle crucial dans la détection des violations de sécurité et la surveillance continue de l'infrastructure pour maintenir un niveau de sécurité optimal.

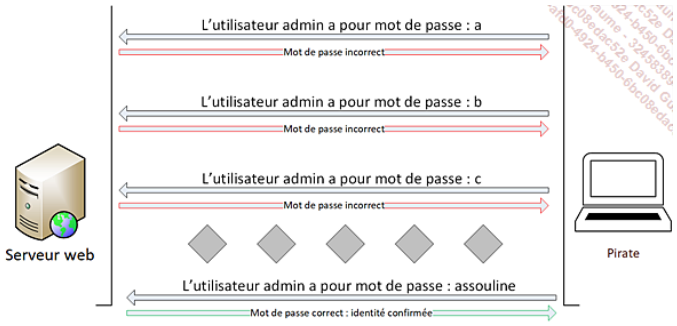
Les principales attaques informatiques

Les principales attaques informatiques comprennent :

1. **Les attaques par reconnaissance** :
    - Recherchent des vulnérabilités dans les défenses du réseau.
    - Les informations obtenues aident à planifier des attaques futures ciblées.
    - Passent souvent inaperçues pour éviter d'éveiller les soupçons des administrateurs.

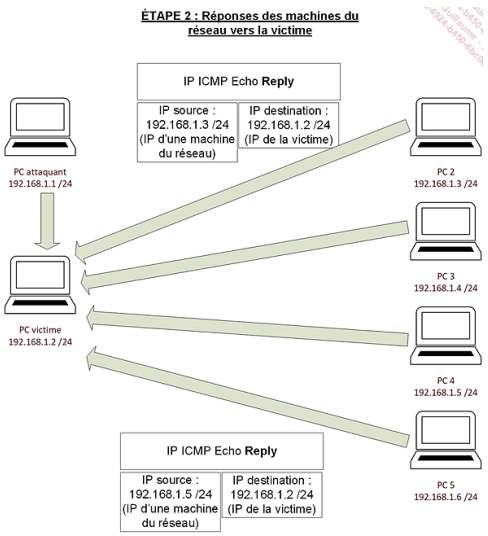
2. **Le Social Engineering** :
    - Exploite la faiblesse humaine en manipulant les individus pour obtenir des informations sensibles.
    - Exemple : se faire passer pour une organisation caritative pour obtenir des informations confidentielles
      .
3. **Le phishing** :
    - Implique l'envoi d'e-mails frauduleux pour inciter les utilisateurs à divulguer leurs informations personnelles.
    - Les e-mails peuvent contenir des liens vers des sites pirates qui captent les données des utilisateurs.

4. **Les attaques par brute force** :
    - Consistent à trouver un mot de passe en testant toutes les combinaisons possibles.
    - Plus le mot de passe est complexe, plus il est difficile à trouver.
    - Des mesures de sécurité telles que la détection des tentatives multiples ou l'utilisation de mots de passe complexes sont nécessaires pour contrer ces attaques.



Les attaques par déni de service

1. **Qu'est-ce qu'une attaque par déni de service** :
   - Elle vise à rendre un service indisponible en saturant ses ressources.
   - Diffère des attaques de reconnaissance qui visent à rester discrètes.
   - Les attaques DDoS (Distributed Denial of Service) utilisent des botnets contrôlés par l'attaquant pour multiplier les attaques.
2. **Attaque par SYN Flood** :
   - Génère des requêtes bogues, souvent à partir de multiples hôtes.
   - Saturation des ressources du serveur ciblé, empêchant les clients légitimes de se connecter.
3. **Attaque par UDP Flooding** :
   - Exploite le protocole UDP et sa priorité sur le TCP, provoquant une congestion du réseau.
   - Utilise des attaques comme le Chargen Denial of Service Attack, exploitant les ports Chargen et Echo.
4. **Attaque de type Smurfing** :
   - Usurpe l'identité de la victime pour envoyer des paquets ICMP Echo Request à l'adresse de broadcast du réseau.
   - Toutes les machines du réseau répondent, surchargeant la victime de trafic entrant et provoquant un déni de service.



Les composants de la cryptographie

Ce chapitre a été traité lors de la SF avec Jean-Luc ; c'est pourquoi, la suite est un bref résumé.

Le principe de fonctionnement du **chiffrement** consiste à transformer un message clair en un message chiffré, rendant ainsi son contenu illisible pour les personnes non autorisées. Ce processus utilise des algorithmes mathématiques et des clés pour sécuriser les données pendant la transmission ou le stockage.

- **Le chiffrement par substitution monoalphabétique** remplace chaque lettre du message clair par une autre lettre selon un seul alphabet de substitution. Cela peut être facilement cassé avec des méthodes telles que l'analyse de fréquence des lettres ou l'indice de coïncidence.

- **Le chiffrement par substitution polyalphabétique** utilise plusieurs alphabets de substitution pour chiffrer le message, rendant plus difficile la cryptanalyse. L'algorithme le plus célèbre est le chiffre de Vigenère, qui utilise une clé pour déterminer quel alphabet de substitution est utilisé à chaque position du message.

Le chiffrement par transposition consiste à réorganiser l'ordre des lettres du message clair selon une méthode spécifique, sans changer les lettres elles-mêmes. Cela peut être réalisé en permutant les lettres selon un schéma défini par une clé.

Les principes du chiffrement symétrique impliquent l'utilisation d'une seule clé pour à la fois chiffrer et déchiffrer les données. Cela garantit la confidentialité des informations entre les parties qui connaissent cette clé.

Le chiffrement asymétrique utilise une paire de clés distinctes: une clé publique et une clé privée. La clé publique est utilisée pour chiffrer les données, tandis que la clé privée correspondante est utilisée pour déchiffrer les données. Cela permet une communication sécurisée sans nécessiter de partage de clés.

Les algorithmes de hash transforment des données de taille variable en une valeur de hachage de taille fixe, généralement sous forme de chaîne hexadécimale. Cela est utilisé pour vérifier l'intégrité des données, créer des signatures numériques et sécuriser les mots de passe.

Les mots de passe sont stockés sous forme de hachage plutôt que sous leur forme brute, ce qui rend difficile pour les attaquants de récupérer les mots de passe d'origine à partir des bases de données compromises.

Les Rainbow Tables sont des bases de données précalculées contenant des paires de valeurs clair-haché, ce qui accélère la recherche inversée de mots de passe. Elles sont souvent utilisées dans les attaques par force brute ou par dictionnaire.

L'algorithme de hachage MD5 est un algorithme largement utilisé mais maintenant considéré comme peu sécurisé en raison de ses vulnérabilités. Il génère une valeur de hachage de 128 bits.

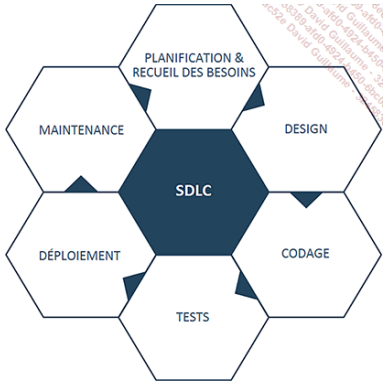
L'algorithme de hachage SHA-1 est également devenu obsolète en raison de ses faiblesses de sécurité, bien qu'il ait été largement utilisé dans le passé. Il produit une valeur de hachage de 160 bits.

Les algorithmes de hash de la famille SHA-2 et SHA-3 sont des améliorations de leurs prédécesseurs en termes de sécurité. Ils génèrent des valeurs de hachage de différentes longueurs, offrant une meilleure résistance aux attaques cryptographiques.

Sécurité du développement et bonnes pratiques

Application de la sécurité au SDLC (Software Development Life Cycle)

Le Software Development Life Cycle (SDLC) intègre la sécurité à chaque étape du processus de développement logiciel, afin de prévenir les vulnérabilités et les cyberattaques. Voici un résumé des principales implications de la sécurité dans chaque phase du SDLC.



1. **Planification & recueil des besoins**
   - Identification des risques liés à la sécurité dès le début du projet.
   - Intégration des exigences de sécurité dans la définition des besoins.
   - Évaluation des risques de sécurité et des critères de succès pour chaque élément du projet.

2. **Conception/design du produit**
   - Élaboration d'une architecture sécurisée à haut niveau (HLD) et à bas niveau (LLD).
   - Définition des mécanismes de sécurité, des contrôles d'accès et des protocoles de chiffrement.
   - Intégration de la sécurité dans la conception des bases de données et des interfaces utilisateur.

3. **Codage et tests**
   - Écriture de code sécurisé en conformité avec les bonnes pratiques de programmation sécurisée.
   - Réalisation de tests de sécurité tout au long du processus de développement, incluant des tests unitaires, d'API, d'intégration, de système, de performance, de capacité et de compatibilité.

- Utilisation d'outils d'analyse statique et dynamique de code pour détecter les vulnérabilités.

4. **Déploiement et maintenance**
   - Intégration continue et déploiement continu pour assurer la sécurité des versions déployées.
   - Surveillance continue des systèmes en production pour détecter et répondre rapidement aux menaces.
   - Application des correctifs de sécurité et des mises à jour régulières pour atténuer les vulnérabilités.

En adoptant une approche SDLC, les organisations peuvent réduire les risques de sécurité tout en améliorant la qualité et la fiabilité de leurs logiciels, évitant ainsi les coûts et les dommages associés aux violations de sécurité.

SSDLC (Secure Software Development Life Cycle)

Le passage du SDLC (Software Development Life Cycle) au SSDLC (Secure Software Development Life Cycle) est essentiel pour répondre aux défis croissants en matière de sécurité informatique. Dans un environnement numérique en constante évolution, où les cybermenaces sont de plus en plus sophistiquées et omniprésentes, intégrer la sécurité dès les premières phases du processus de développement est devenu impératif. Le SSDLC reconnaît que la sécurité des applications ne peut pas être une réflexion après-coup, mais doit être une priorité tout au long du cycle de vie du logiciel. En adoptant le SSDLC, les organisations peuvent anticiper les vulnérabilités, minimiser les risques de sécurité et réduire les coûts associés à la correction de failles découvertes tardivement. De plus, en impliquant les équipes de développement dans la prise en compte des aspects de sécurité dès le début du processus, le SSDLC favorise une culture de sécurité proactive et collaborative au sein de l'organisation, améliorant ainsi la confiance des utilisateurs et la réputation de l'entreprise. En fin de compte, le SSDLC offre une approche holistique qui garantit que la sécurité est intégrée de manière transparente à chaque étape du développement logiciel, offrant ainsi une protection renforcée contre les menaces numériques en constante évolution.

➔ C’est la vision DevSecOps

1. **Mise en œuvre du Threat Modeling**
   - Le Threat Modeling (modélisation des menaces) consiste à identifier, mesurer et évaluer les menaces potentielles auxquelles un produit peut être confronté.
   - Les équipes réalisent généralement des schémas pour mettre en lumière les failles et les surfaces d'attaque potentielles.
   - Il existe un Manifeste de Threat Modeling qui encourage une approche systémique, la créativité et la pluralité des points de vue.
   - Cinq grands périmètres d'actions sont à prendre en compte, notamment l'intégration de toutes les parties prenantes, l'application du Threat Modeling à toute l'organisation et l'utilisation d'outils innovants.

2. **Utilisation du modèle STRIDE**

- Le modèle STRIDE est simple d'utilisation et permet d'identifier les surfaces d'attaque liées aux vulnérabilités avant même le développement.
- Il se compose de six éléments : Spoofing Identity, Tampering with Data, Repudiation, Information Disclosure, Denial of Service et Elevation of Privilege.
- Chaque élément du modèle correspond à une menace potentielle, par exemple, l'usurpation d'identité, la modification non autorisée des données, etc.

3. Contraintes de sécurité en Agilité:

- Dans le développement Agile, il peut y avoir une tension entre la livraison rapide de nouvelles fonctionnalités et la sécurité.
- Il est crucial d'évaluer le niveau d'importance de la disponibilité, de l'intégrité et de la confidentialité des besoins.
- La liste des événements redoutés est dressée en fonction des risques potentiels pour la sécurité.
- Les équipes rédigent des Abuse User Stories pour se mettre à la place de l'attaquant et identifier les vulnérabilités.
- Des tâches sont alors construites pour atténuer les risques associés à ces scénarios d'abus.

4. Architecture d'une application Cloud sécurisée :

- Menaces et responsabilité partagée :
    - Dans les environnements Cloud, les responsabilités en matière de sécurité sont partagées entre le fournisseur de service Cloud et le client, en fonction du type de service utilisé (SaaS, PaaS, IaaS).
    - Les principales vulnérabilités restent liées aux erreurs humaines, notamment l'accès non autorisé à la console du Cloud.
    - Les clés d'accès aux API sont une cible principale pour les attaquants, donc il est crucial de mettre en place une surveillance et un journalisation adéquates.
- Bénéfices du Cloud pour la sécurité des applications :
    - Les services Cloud offrent des avantages en matière de sécurité, tels que la gestion des accès (IAM), la possibilité de déployer rapidement et à l'échelle, ainsi que des solutions de chiffrement pour les données.
    - Plus l'utilisation de services managés est importante (SaaS, PaaS), moins les préoccupations en matière de sécurité sont importantes, mais cela signifie également un contrôle moindre sur la gestion de la sécurité des applications.

5. Utilisation de différents environnements pour sécuriser le déploiement :

- Notions générales autour des environnements :
    - Les environnements de développement, d'intégration continue, de préproduction et de production sont utilisés pour tester et déployer les applications.
    - Chaque environnement a ses propres objectifs et permet de vérifier différentes étapes du processus de développement et de déploiement.
- Exemple avec une application web :
    - Les développeurs travaillent sur leur code dans un environnement de développement, puis les modifications sont testées via un pipeline

d'intégration continue avant d'être déployées dans les environnements de préproduction et de production.
    - Les environnements de préproduction et de production sont utilisés pour tester les fonctionnalités et la performance de l'application avant sa mise à disposition des clients.

Le TOP 10 de l'OWASP

Le Top 10 de l'OWASP (Open Web Application Security Project) est une liste des dix principales vulnérabilités de sécurité des applications Web, identifiées et publiées régulièrement par l'OWASP, une organisation à but non lucratif dédiée à l'amélioration de la sécurité des logiciels.

1. **Injection SQL**
   Une faille qui permet à un attaquant d'injecter des requêtes SQL malveillantes dans une application, ce qui peut compromettre la base de données sous-jacente.

2. **Cross-Site Scripting (XSS)**
   Une attaque où un attaquant injecte du code malveillant (généralement du JavaScript) dans des pages Web consultées par d'autres utilisateurs, permettant à l'attaquant de voler des informations ou de prendre le contrôle des sessions utilisateur.

3. **Broken Authentication**
   Des failles dans le processus d'authentification et de gestion des sessions qui peuvent permettre à un attaquant de compromettre les comptes utilisateur, les mots de passe, les jetons d'authentification, etc.

4. **Sensitive Data Exposure**
   La mauvaise protection ou la divulgation involontaire de données sensibles telles que les informations d'identification, les numéros de carte de crédit, etc.

5. **XML External Entities (XXE)**
   Une vulnérabilité dans les applications Web qui permet à un attaquant d'inclure des entités XML externes malveillantes, pouvant entraîner une fuite d'informations confidentielles ou d'autres attaques.

6. **Broken Access Control**
   Des contrôles d'accès inadéquats qui permettent à un utilisateur non autorisé d'accéder à des fonctionnalités ou à des données auxquelles il ne devrait pas avoir accès.

7. **Security Misconfiguration**
   La mauvaise configuration des paramètres de sécurité, des serveurs, des applications, etc., qui peuvent rendre l'application vulnérable aux attaques.

8. **Cross-Site Request Forgery (CSRF)**
   Une attaque où un attaquant force un utilisateur authentifié à effectuer des actions indésirables sur une application Web sans son consentement.

9. **Using Components with Known Vulnerabilities**
   L'utilisation de composants logiciels obsolètes ou connus pour être vulnérables, ce qui expose l'application à des risques de sécurité.

10. **Insufficient Logging & Monitoring**
    Une mauvaise gestion des journaux d'activité et du suivi des événements de sécurité, ce qui peut entraîner un manque de visibilité sur les attaques et les violations de sécurité.

Ces vulnérabilités représentent les principales menaces auxquelles les applications Web sont confrontées, et leur identification et leur correction sont essentielles pour assurer la sécurité des applications en ligne.

Gestion des évènements de votre infrastructure

La mise en place d'un système de log est essentiel dans l'infrastructure informatique.

- **Traçabilité**
  Les logs fournissent une trace des activités et des événements dans un système, permettant de détecter et d'investiguer les incidents de sécurité.
- **Analyse des menaces**
  Ils offrent une source d'information précieuse pour comprendre les schémas d'attaques et renforcer les défenses en identifiant les comportements malveillants.
- **Conformité**
  Les logs facilitent la conformité aux normes et réglementations en matière de sécurité en documentant les actions effectuées, les accès aux données sensibles, et les changements de configuration, essentiels pour les audits de sécurité et les exigences légales.

Bibliographie

**Les images non référencées dans la bibliographie proviennent du livre étudié.*