Notion de risques : Chap.6.b : Sécurité des logiciels : Part.1

mardi 20 février 2024 15:01

Chap.6.b : Sécurité des logiciels (Focus Architecture WEB N1, N2, N3)

- SDLC dans une Architecture WEB N1, N2, N3)
  - Faille de sécurité à intégrer dans le cycle de développement logiciel
  - Sécurité dans le cycle de mise en production des applications Web

A. SDLC dans une Architecture WEB N1, N2, N3

1. Faille de sécurité à intégrer dans le cycle de développement logiciel (SDLC) pour les applications Web

Ne pas oublier que dans le contexte du développement d'applications Web utilisant PHP et une base de données MySQL, les considérations suivantes doivent être traitées :

1. **Analyse des Risques dès le Départ :** Lors de la phase de conception, il est essentiel d'identifier les risques de sécurité potentiels liés à l'application, tels que les injections SQL, les attaques XSS (Cross-Site Scripting) ou les failles d'authentification.

2. **Validation et Échappement des Données :** Tout input utilisateur, notamment les données provenant de formulaires Web, doit être validé et échappé correctement pour prévenir les attaques d'injection SQL et XSS. Les fonctions de validation intégrées à PHP et les requêtes préparées peuvent être utilisées pour sécuriser les interactions avec la base de données MySQL.

3. **Gestion des Sessions et des Cookies :** Les sessions doivent être gérées de manière sécurisée pour éviter les attaques de fixation de session et de vol de session. Il est recommandé d'utiliser les fonctions de gestion de session de PHP, telles que `session_start()` et `session_regenerate_id()`, et de sécuriser les cookies en définissant des attributs appropriés, comme le flag `HttpOnly` pour les cookies sensibles.

4. **Protection contre les Attaques CSRF (Cross-Site Request Forgery) :** Les formulaires Web doivent être protégés contre les attaques CSRF en utilisant des jetons anti-CSRF (CSRF tokens) générés de manière aléatoire et vérifiés lors de la soumission des formulaires.

5. **Mises à Jour Régulières et Sécurité du Code :** Il est crucial de maintenir le code de l'application à jour en appliquant les correctifs de sécurité publiés par les développeurs de PHP et MySQL. De plus, l'utilisation de bonnes pratiques de codage sécurisé, telles que le principe du moindre privilège et l'évitement des fonctions obsolètes, contribue à réduire les vulnérabilités.

2. Sécurité dans le cycle de mise en production des applications Web

La sécurisation du cycle de mise en production des applications Web revêt une importance cruciale pour les informaticiens chargés du déploiement et de la gestion des systèmes. Voici les processus et les mesures de sécurité spécifiques à mettre en œuvre, en tenant compte des architectures N1, N2 et N3 :

Processus de Mise en Production

1. **Déploiement depuis l'Architecture de Développement (N1) :** Une fois le développement

terminé sur l'architecture N1, les informaticiens utilisent des outils de gestion de versions tels que **Git** pour transférer le code vers l'architecture d'intégration (N2).

2. **Tests d'Intégration (N2) :** Sur l'architecture N2, les informaticiens utilisent des outils de gestion des tests tels que **PHPUnit** pour effectuer des tests d'intégration. Des plates-formes d'intégration continue comme **Jenkins** peuvent être utilisées pour automatiser ce processus.

3. **Validation des Tests d'Intégration :** Après la réussite des tests d'intégration, les informaticiens effectuent une revue de code et une validation finale sur l'architecture N2. Ils utilisent des outils de revue de code comme **Gerrit** pour garantir la qualité et la sécurité du code avant le déploiement en production.

4. **Déploiement en Production (N3) :** Lors du déploiement en production sur l'architecture N3, les informaticiens utilisent des outils de gestion de configuration tels que **Ansible** pour automatiser le déploiement et garantir la cohérence de l'infrastructure.

Tests à Effectuer

1. **Tests de Performance :** Les informaticiens utilisent des outils de test de charge comme **Apache JMeter** pour évaluer les performances de l'application sous différentes conditions de charge et s'assurer qu'elle répond aux exigences de performance.

2. **Tests de Sécurité :** Des outils d'analyse de sécurité automatisés tels que **OWASP ZAP** sont utilisés pour identifier et corriger les vulnérabilités de l'application. Les informaticiens effectuent également des **tests manuels de sécurité** pour détecter les failles de sécurité complexes.

3. **Tests de Disponibilité :** Les informaticiens utilisent des outils de surveillance comme **Nagios** pour surveiller la disponibilité de l'application en production. Des tests de basculement et de reprise sur sinistre sont effectués pour garantir une disponibilité continue.

Sécurité à Mettre en Œuvre

1. **Gestion des Accès et des Identités :** Les informaticiens mettent en œuvre une gestion rigoureuse des accès et des identités à l'aide d'**outils d'authentification centralisée comme LDAP**. Ils utilisent des outils de **gestion des droits d'accès** comme **Keycloak** pour attribuer des privilèges aux utilisateurs en fonction de leurs rôles.

2. **Surveillance et Journalisation :** Les informaticiens utilisent des outils de surveillance et de journalisation comme **ELK Stack** (Elasticsearch, Logstash, Kibana) pour collecter, analyser et visualiser les journaux d'audit et les événements de sécurité en temps réel.

3. **Automatisation de la Sécurité :** L'automatisation des tâches de sécurité est essentielle pour garantir une réponse rapide aux menaces. Les informaticiens utilisent des outils d'automatisation comme Ansible et Puppet pour appliquer automatiquement les correctifs de sécurité et les configurations recommandées.

Processus de Sécurisation pour les Informaticiens dans les 3 Architectures

1. Architecture N1 (Développement)

- **Accès au Réseau de Développement :**
  - Les développeurs doivent être autorisés à accéder au réseau de développement uniquement via des identifiants sécurisés.
  - L'accès au serveur de développement doit être limité aux seuls informaticiens autorisés, avec des connexions SSH sécurisées pour les transferts de fichiers et l'administration.

- **Sécurisation des Outils de Développement :**
  - Les outils de développement tels que Visual Studio doivent être régulièrement mis à jour avec les derniers correctifs de sécurité.
  - Les environnements de développement intégrés (IDE) doivent être configurés avec des paramètres de sécurité appropriés pour prévenir les vulnérabilités.

- **Processus de Développement Sécurisé :**
  - Les développeurs doivent suivre des pratiques de développement sécurisé, notamment en échappant les entrées utilisateur, en validant les données et en évitant les vulnérabilités courantes telles que les injections SQL et XSS.
  - Des analyses de sécurité du code doivent être effectuées régulièrement pour identifier et corriger les failles de sécurité potentielles.

2. Architecture N2 (Intégration)

- **Accès au Réseau d'Intégration :**
  - Les administrateurs système et les développeurs doivent être autorisés à accéder au réseau d'intégration via des identifiants sécurisés et des connexions VPN si nécessaire.
  - Les accès doivent être strictement contrôlés et limités aux personnes nécessaires à l'intégration et aux tests.

- **Sécurisation des Outils d'Intégration :**
  - Les outils d'intégration continue (CI) et de gestion de versions comme Git doivent être configurés avec des contrôles d'accès appropriés pour limiter l'accès au code source et aux artefacts d'intégration.
  - Des mécanismes de surveillance doivent être mis en place pour détecter toute activité suspecte ou non autorisée sur les outils d'intégration.

- **Processus d'Intégration Sécurisée :**
  - Les environnements d'intégration doivent être isolés du reste du réseau pour minimiser les risques de compromission.
  - Des tests de sécurité automatisés doivent être effectués à chaque intégration pour identifier les problèmes de sécurité potentiels dès que possible.

3. Architecture N3 (Production)

- **Accès au Réseau de Production :**
  - L'accès au réseau de production doit être strictement restreint aux administrateurs système autorisés et au personnel de support technique via des connexions sécurisées et des VLANs dédiés.
  - Les accès doivent être surveillés et audités régulièrement pour détecter toute activité suspecte.

- **Sécurisation des Outils de Production :**
  - Les outils de surveillance et de gestion des configurations doivent être configurés avec des niveaux appropriés d'accès et de contrôle pour protéger les systèmes de production contre les modifications non autorisées.
  - Des procédures de sauvegarde régulières doivent être mises en place pour assurer la disponibilité des données et des applications en cas de sinistre.

- **Processus de Gestion de la Production :**
  - Les mises à jour de sécurité doivent être appliquées régulièrement sur tous les systèmes de production pour corriger les vulnérabilités connues.
  - Des tests de pénétration et des audits de sécurité doivent être effectués périodiquement pour évaluer l'efficacité des mesures de sécurité en place et identifier les éventuelles lacunes.

3. Précisions sur les outils mentionnés :

1. **PHPUnit** : PHPUnit est un framework de test unitaire pour le langage de programmation PHP. Il permet aux développeurs de créer des tests automatisés pour vérifier le bon fonctionnement de leurs applications PHP. PHPUnit offre des fonctionnalités avancées telles que la création de jeux de données de test, l'assertion des résultats attendus et la génération de rapports détaillés sur les tests exécutés.

2. **Jenkins** : Jenkins est un outil d'intégration continue open source largement utilisé dans le développement logiciel. Il permet aux équipes de développement de créer des pipelines d'intégration continue pour automatiser les processus de construction, de test et de déploiement des applications. Jenkins offre une grande flexibilité grâce à son écosystème de plugins, ce qui permet aux équipes de personnaliser leur pipeline en fonction de leurs besoins spécifiques.

3. **Gerrit** : Gerrit est une plateforme de revue de code open source conçue pour les projets basés sur Git. Il permet aux développeurs de soumettre leur code pour examen par leurs pairs, de commenter les changements proposés et de suggérer des améliorations. Gerrit offre des fonctionnalités avancées telles que la gestion des permissions, la notification automatique des révisions et l'intégration avec des outils de build et de test.

4. **Ansible** : Ansible est un outil d'automatisation open source utilisé pour la gestion de la configuration, le déploiement d'applications et l'orchestration des infrastructures. Il permet aux équipes informatiques de déployer et de gérer des environnements complexes de manière efficace et reproductible en utilisant des playbooks YAML pour décrire les tâches à effectuer. Ansible est apprécié pour sa simplicité d'utilisation, sa facilité de déploiement et sa grande extensibilité.

5. **OWASP ZAP** : OWASP ZAP (Zed Attack Proxy) est un outil de test de sécurité automatisé utilisé pour identifier et corriger les vulnérabilités de sécurité dans les applications web. Il effectue des analyses de sécurité automatisées en simulant des attaques de type injection SQL, cross-site scripting (XSS), et d'autres types d'attaques courantes. OWASP ZAP offre une interface graphique conviviale ainsi que des fonctionnalités avancées pour les tests de sécurité.

6. **MySQL Workbench** : MySQL Workbench est un outil de modélisation de base de données et d'administration pour MySQL. Il permet aux développeurs et aux administrateurs de base de données de concevoir, de visualiser et de gérer des bases de données MySQL à l'aide d'une interface graphique intuitive. MySQL Workbench offre des fonctionnalités telles que la conception de schémas de base de données, la gestion des utilisateurs et des privilèges, et l'exécution de requêtes SQL.