LISV Laboratoire d'ingénierie des systèmes de Versailles
UVSQ université PARIS-SACLAY

Programmation sécurisée (Secure coding)
TP 5
OWASP SAMM

Halim Djerroud

révision 1.0

Contexte

SpeedSoft est une entreprise spécialisée dans le développement de solutions logicielles destinées aux entreprises. Son produit phare est une application collaborative de gestion des tâches, conçue pour répondre aux besoins d'équipes de taille petite à moyenne. Cette application permet de créer, suivre et partager des tâches de manière efficace.

Récemment, plusieurs clients ont exprimé des inquiétudes au sujet de la sécurité de l'application, suite à la publication de rapports médiatiques évoquant des failles similaires dans des produits concurrents. Les principales préoccupations identifiées incluent :

- La protection des données sensibles (par exemple, les mots de passe et les informations des utilisateurs).
- La prévention des attaques courantes, telles que les injections SQL et les failles XSS.
- L'absence de processus formels garantissant une sécurité continue tout au long du cycle de vie du développement logiciel.
- La conformité aux réglementations, notamment le RGPD pour l'Europe et le HIPAA pour les clients internationaux américains.

Actuellement, l'équipe technique utilise un processus de développement logiciel agile, mais ne s'appuie sur aucun cadre formel de sécurité. Les tests de sécurité sont rares et réalisés uniquement en fin de projet. Par ailleurs, aucun outil automatisé n'est employé pour l'analyse du code ou la détection des vulnérabilités.

Face à ces défis, l'entreprise souhaite intégrer le modèle **OWASP SAMM** pour évaluer ses pratiques de sécurité actuelles et les améliorer. Votre mission consiste à aider *SpeedSoft* à identifier les lacunes dans ses processus, en vous basant sur le modèle **OWASP SAMM**, et à proposer des recommandations concrètes et un plan d'action réalisable, adapté à leur flux de travail agile.

Enfin, il est important de noter que l'entreprise développe principalement des applications de gestion des tâches personnalisées, en fonction des cahiers des charges spécifiques de chaque client. Ces variations impliquent des implémentations adaptées aux besoins particuliers des différents clients.

Voici un aperçu des principales implémentations réalisées :

- https://github.com/hdd-robot/ToDoListBack
- https://github.com/hdd-robot/projet_web_jira_like
- https://github.com/hdd-robot/gestion-de-tache
- https://github.com/hdd-robot/web-project-task-management
- https://github.com/hdd-robot/EventSync
- https://github.com/hdd-robot/TaskOptimizer
- https://github.com/hdd-robot/XDEV5-TODOLIST
- https://github.com/hdd-robot/web-project
- https://github.com/hdd-robot/Back-Web-Final-Fred-Choco-SL

Cahier des charges générique

Le projet vise à développer une application web de gestion de tâches permettant aux utilisateurs de créer, organiser et suivre leurs tâches de manière efficace. Chaque tâche disposera d'attributs spécifiques tels qu'un titre, une description, un projet d'appartenance, une date de création, une date d'échéance, une priorité, un statut, des tags, et d'autres informations personnalisables. De plus, une tâche pourra être décomposée en sous-tâches avec une hiérarchie de sous-niveaux illimitée.

L'application offrira les fonctionnalités suivantes :
- Création de tâches : Les utilisateurs pourront créer des tâches en y ajoutant un titre, une description, une priorité, une échéance, des tags, et d'autres informations pertinentes.
- Organisation des tâches : Les tâches pourront être classées et affichées par projet, liste, tag ou date d'échéance, offrant une vue personnalisée adaptée aux besoins des utilisateurs.
- Mise à jour et suppression : Les utilisateurs auront la possibilité de modifier ou de supprimer les informations liées à une tâche, selon les besoins.
- Attribution des tâches : Les tâches pourront être assignées à d'autres utilisateurs, facilitant la gestion collaborative.
- Suivi de l'avancement : Chaque tâche disposera d'un statut évolutif permettant de suivre son avancement (par exemple : "À faire", "En cours", "Terminée").
- Notifications : Les utilisateurs recevront des notifications, notamment par email, en cas d'attribution d'une tâche, de modification ou d'approche d'une échéance.
- Collaboration en temps réel : L'application permettra aux utilisateurs de travailler simultanément sur des tâches partagées, favorisant la collaboration au sein des équipes.

Instructions

Partie 1 : Compréhension et Analyse

1. **Lecture du contexte**
   - Lire la description du modèle OWASP SAMM.
   - Se familiariser avec les cinq domaines de SAMM :
     (a) Gouvernance
     (b) Conception
     (c) Implémentation
     (d) Vérification
     (e) Opérations
   - Se familiariser avec la toolbox SAMM (*SAMM spreadsheet.xlsx*) :
2. **Identification des lacunes**
   - Étudiez les pratiques actuelles de l'entreprise (document fourni et code source).
   - Repérez **trois domaines** où des améliorations sont nécessaires (ex. : manque de revue de code, tests de sécurité insuffisants, absence de politiques de gouvernance).
3. **Élaboration d'un plan d'amélioration**
   - Pour chaque domaine identifié, proposez :
     - Une action immédiate à mettre en œuvre.
     - Un objectif à moyen terme.
     - Les ressources ou outils nécessaires.

Partie 2 : Mise en Pratique

1. **Étude de cas : Révision de code**
   - Un extrait de code sera fourni (par exemple, une fonction de gestion des mots de passe).
   - Analysez ce code à l'aide des principes OWASP (ex. : Injection SQL, stockage sécurisé des mots de passe).
   - Identifiez et documentez les vulnérabilités potentielles.
   - Proposez une version corrigée du code.
2. **Simulation : Planification d'un test de sécurité**
   - Concevez un plan de test pour vérifier la sécurité de l'application Web.

- Le plan doit inclure :
  - Les types de tests (ex. : tests statiques, dynamiques).
  - Les outils à utiliser (ex. : ZAP, Burp Suite).
  - Un calendrier pour ces tests.

Livrables attendus

1. Une analyse des lacunes dans les pratiques actuelles.
2. *SAMM spreadsheet.xlsx* complété dans la mesure du possible.
3. Un plan d'amélioration basé sur OWASP SAMM.
4. Une version corrigée de l'extrait de code.
5. Un plan de test détaillé pour la sécurité de l'application.