Ecol

Ecole des Mines de l'Industrie et de la Géologie

EVALUATION (DUREE 2H) — CORRIGÉ

Partie I : Questions de compréhension (6 points)

1. Définissez la sécurité web. Citez deux objectifs principaux. (2 pts)

La sécurité web est l'ensemble des mesures, techniques et bonnes pratiques mises en œuvre pour protéger les applications et sites web contre les attaques, les accès non autorisés, le vol ou la corruption de données.
Deux objectifs principaux :
- Garantir la confidentialité, l'intégrité et la disponibilité des données (triade CIA).
- Protéger les utilisateurs et le système contre les attaques (injections, XSS, vol de session, etc.) et assurer la continuité de service.

2. Expliquez ce que signifie « OWASP Top 10 ». Pourquoi est-ce important ? (1 pts)

L'OWASP Top 10 est un classement, publié par l'organisation OWASP (Open Web Application Security Project), des dix vulnérabilités de sécurité les plus critiques et les plus fréquentes dans les applications web (ex : injection, XSS, mauvaise gestion de l'authentification...).
C'est important car il sert de référence mondiale pour sensibiliser les développeurs, prioriser les efforts de sécurisation et orienter les tests de sécurité des applications.

3. Donnez deux exemples d'outils utilisés pour tester la sécurité des applications web. (0.5 pts)

OWASP ZAP et Burp Suite (d'autres exemples valables : Nmap, Nikto, sqlmap).

4. Citez quatre bonnes pratiques de sécurité à appliquer dans le développement web. (2 pts)

- Valider et filtrer toutes les entrées utilisateur côté serveur.
- Utiliser des requêtes préparées (paramétrées) pour éviter les injections SQL.
- Hacher les mots de passe avec un algorithme robuste comme bcrypt.
- Utiliser HTTPS pour chiffrer les échanges et sécuriser les cookies (HttpOnly, Secure).

5. Quelle est la différence entre une norme et une réglementation ? Donnez un exemple de chaque. (0.5 pts)

Une norme est un référentiel de bonnes pratiques d'application volontaire (ex : ISO/IEC 27001), tandis qu'une réglementation est un texte légal obligatoire imposé par une autorité, dont le non-respect entraîne des sanctions (ex : le RGPD).

Partie II : QCM (10 points)

1. Parmi les éléments suivants, lesquels relèvent d'une attaque côté client ? (1 pts)

- ■ Brute Force
- ■ Cross-Site Scripting

- ■ Injection SQL

2. Le protocole HTTPS permet : (1 pts)

- ■ De chiffrer les échanges entre client et serveur
- ■ D'empêcher toutes les attaques XSS
- ■ De garantir l'intégrité des données transmises

3. Vrai ou Faux : (1 pts)

a) Les mots de passe doivent être stockés en clair dans la base de données. → **FAUX**

b) Une session doit être invalidée à la déconnexion de l'utilisateur. → **VRAI**

4. Cochez les bonnes affirmations : (1 pts)

- ■ Un test d'intrusion peut être réalisé par un auditeur externe
- ■ Un scanner de vulnérabilités recherche automatiquement des failles connues
- ■ Les tests de sécurité sont uniquement effectués après la mise en production

5. Vrai ou Faux : (1 pts)

a) Un outil comme OWASP ZAP permet de faire des scans actifs d'une application web. → **VRAI**

b) Les tests de sécurité ne concernent que les développeurs backend. → **FAUX**

6. Vrai ou faux : (1 pts)

a) Le standard OWASP Top 10 est une réglementation légale européenne. → **FAUX**

b) PCI-DSS est obligatoire pour les systèmes qui traitent des paiements par carte bancaire. → **VRAI**

7. Cochez les bonnes réponses : (1 pts)

- ■ Les mots de passe doivent être hachés avec un algorithme adapté comme bcrypt
- ■ Il est préférable d'informer l'utilisateur si son identifiant est incorrect même si son mot de passe est juste
- ■ Les paramètres utilisateurs doivent toujours être validés côté serveur
- ■ HTTPS protège automatiquement contre toutes les attaques web

8. Vrai ou faux : (1 pts)

a) Il est conseillé d'utiliser des requêtes préparées pour se protéger contre les injections SQL. → **VRAI**

b) Une API REST sécurisée ne nécessite pas d'authentification si elle est sur un réseau privé. → **FAUX**

9. Cochez les affirmations vraies (1 pts)

- ■ Une injection SQL est inoffensive si l'application est en lecture seule
- ■ Une faille XSS permet l'exécution de scripts dans le navigateur de la victime
- ■ Un CSRF peut être évité uniquement en utilisant HTTPS
- ■ Un cookie sans l'attribut HttpOnly peut être volé via du JavaScript malveillant

10. Cochez les affirmations vraies : (1 pts)

- ■ Le RGPD s'applique aux entreprises hors UE si elles traitent des données de citoyens européens
- ■ Une adresse IP n'est pas considérée comme une donnée personnelle
- ■ La norme ISO/IEC 27001 concerne la sécurité des systèmes d'information
- ■ Le RGPD impose de chiffrer toutes les bases de données

Partie III – Analyse d'un code vulnérable (4 points)

On vous donne le code PHP suivant pour traiter une page de connexion utilisateur :

```
<?php
    $conn = mysqli_connect("localhost", "root", "", "app");
    $user = $_POST['username'];
    $pass = $_POST['password'];
    $query = "SELECT * FROM users WHERE username = '$user' AND password =
'$pass'";
    $result = mysqli_query($conn, $query);
?>
```

a) Identifiez la vulnérabilité présente dans ce code. Donnez un exemple d'exploitation. (2 pts)

Le code présente une vulnérabilité d'injection SQL : les variables $user et $pass, issues directement de $_POST, sont insérées dans la requête SQL sans aucune validation, échappement ou requête préparée.
Exemple d'exploitation : un attaquant peut saisir comme identifiant `' OR '1'='1' --` *et un mot de passe quelconque. La requête générée devient alors toujours vraie, ce qui permet de contourner l'authentification et de se connecter sans connaître de mot de passe valide.*

b) Proposez une solution pour corriger cette faille. (2 pts)

Utiliser des requêtes préparées (paramétrées) avec mysqli ou PDO, qui séparent le code SQL des données saisies par l'utilisateur, par exemple :

```
$stmt = $conn->prepare("SELECT * FROM users WHERE username = ? AND password = ?");
$stmt->bind_param("ss", $user, $pass);
$stmt->execute();
```

Il faut également stocker les mots de passe sous forme hachée (ex : bcrypt via password_hash / password_verify) plutôt qu'en clair, et valider/filtrer les entrées côté serveur.