

Les principales failles de sécurité

Table des matières

I. Contexte	3
II. Failles de sécurité communes	3
A. Concept de faille de sécurité.....	3
B. Cross-Site Scripting	4
C. CSRF	7
D. Exercice : Quiz.....	9
III. Autres failles de sécurité	10
A. Les failles de Base de donnée	10
B. Autres failles	13
C. Veilles et métier	15
D. Exercice : Quiz.....	16
IV. L'essentiel	17
V. Auto-évaluation	17
A. Exercice	17
B. Test.....	18
Solutions des exercices	20

I. Contexte

Durée : 1 h

Pré-Requis : base en code

Environnement de travail : navigateur web / Repl.it

Contexte

La sécurité dans le développement est une préoccupation majeure due à la numérisation croissante des activités. Les applications web sont exposées à des failles de sécurité qui peuvent compromettre la confidentialité des données.

Dans ce cours, nous chercherons à aborder les enjeux de sécurité et intégrer l'importance de cette dernière.

Nous aborderons les failles les plus courantes, à savoir le Cross-Site Scripting (XSS) et le Cross-Site Request Forgery (CSRF), ainsi que les failles touchant les bases de données, l'inclusion de fichiers et l'authentification insuffisante.

Nous verrons également l'importance de la veille technologique et du métier de pentester. Ce cours vous permettra de connaître les failles de sécurité principales et les protections pouvant être mise en place.

II. Failles de sécurité communes

A. Concept de faille de sécurité

Définition

Faille de sécurité

Une faille de sécurité, également connue sous le nom de vulnérabilité, est une lacune ou une faiblesse présente dans le code d'une application ou d'un système qui pourrait être exploitée par des personnes mal intentionnées, pour accéder, manipuler ou endommager des données sensibles et privées.

Risque des failles de sécurités

Ces vulnérabilités peuvent exister pour diverses raisons, que ce soit des erreurs de programmation, une configuration incorrecte, des contrôles de sécurité insuffisants, voire l'utilisation de technologies obsolètes. Elles peuvent permettre à des attaquants d'exécuter du code, d'accéder à des informations sensibles, de modifier le comportement du système, de l'arrêter, voire d'en prendre le contrôle total (de façon visible ou invisible).

Les failles de sécurité comportent plusieurs formes, par exemple l'injection SQL, le Cross-Site Scripting (XSS), le Cross-Site Request Forgery (CSRF), etc. C'est pour cette raison que la sécurisation des applications doit être une préoccupation majeure en développement web et dans tous les autres domaines de la technologie.

Ces dernières peuvent avoir des répercussions majeures pour une entreprise, affectant sa réputation mais aussi ses résultats financiers. Lorsqu'une faille est exploitée, les informations sensibles de l'entreprise, qu'il s'agisse de données client, de plans stratégiques ou de propriété intellectuelle, peuvent être compromises.

Il y a aussi un risque d'entraîner des pertes financières directes, par exemple si des données bancaires ou de carte de crédit sont volées et utilisées frauduleusement.

Les entreprises peuvent également être tenues légalement responsables si elles ne sont pas à protéger les données de leurs clients en respectant les règles de sécurité, entraînant des amendes importantes et des actions en justice.

À long terme, une faille de sécurité peut éroder la confiance des clients de l'entreprise, affectant ainsi les relations et les revenus potentiels à venir ou les partenariats.

Les conséquences sont donc principalement subies sur 3 aspects :

- Financière directe ou indirecte
- Légale
- Confiance des clients et partenaires

Conseil L'importance du security by design

L'approche « *Security by Design* » s'est vite démarquée dans le monde du développement en raison de son objectif principal : intégrer la sécurité à chaque étape du processus, plutôt que de l'ajouter après différente phase.

Cette approche permet de s'assurer que les applications sont conçues dès le départ avec une sécurité robuste et fiable. Ainsi, les vulnérabilités sont identifiées et corrigées avant le déploiement, réduisant ainsi le risque de failles de sécurité. Ce dernier met aussi l'accent sur la prise en compte de la sécurité lors de la conception des fonctionnalités, permettant de valider que les exigences de sécurité ne sont pas en conflit avec la fonctionnalité ou la facilité d'utilisation du logiciel (par exemple, une connexion en 30 étapes serait bien sécurisée mais compliquerait énormément l'utilisation).

Le « *Security by Design* » contribue, alors, à protéger les applications des cyberattaques, à instaurer la confiance des utilisateurs ou clients et à respecter les réglementations en matière de protection des données (notamment la RGPD), tout en évitant d'élevé les coûts associés à la correction des failles de sécurités après le déploiement.

B. Cross-Site Scripting

Explication de la faille XSS

La faille XSS, ou Cross-Site Scripting, est un type d'attaque courante dans le développement web, elle consiste en un attaquant injectant du code malveillant, généralement du JavaScript, dans une page web.

Cette page, lorsqu'elle est visualisée par un utilisateur, exécute le code malveillant, ce qui peut avoir diverses conséquences et cibles, allant du vol de cookies jusqu'à la prise de contrôle du compte de l'utilisateur.

La faille XSS se produit le plus souvent lorsque les applications acceptent les entrées des utilisateurs sans les valider correctement, permettant ainsi l'injection de code.

Il existe trois types principaux de failles XSS :

- Stockées : elle se produit lorsque le code malveillant est stocké de manière permanente sur le serveur.
- Reflétées : exécute le code malveillant à partir d'un lien URL.
- DOM-based : a lieu lorsque le code JavaScript malveillant manipule le Document Object Model (DOM) d'une page pour exécuter le code malveillant.

La protection contre les failles XSS nécessite la mise en œuvre de pratiques de codage sûres, comme la validation des entrées par le serveur, l'encodage des sorties et l'utilisation de politiques de sécurité de contenu.

Cette faille est documentée par l'OWASP à cette adresse (lien en anglais) : <https://owasp.org/www-community/attacks/xss>.

Fonctionnement de la faille XSS

L'attaque Cross-Site Scripting (XSS) est un type d'attaque dans laquelle un attaquant injecte du code JavaScript malveillant dans une page Web qui est ensuite exécutée par l'utilisateur final. Pour que cette attaque soit réussie, l'application doit inclure des données non filtrées fournies par l'utilisateur dans une page Web via notamment les inputs.

Prenons un exemple concret pour illustrer comment cela fonctionne.

Imaginons que nous ayons un forum de discussion en ligne où les utilisateurs peuvent poster des commentaires. Un utilisateur malveillant pourrait alors écrire un commentaire contenant du code JavaScript, par exemple : `<script>stealCookies()</script>`, où `stealCookies()` est une fonction hypothétique qui vole les cookies du navigateur de l'utilisateur.

Si le site Web n'a pas été sécurisé pour prévenir les attaques XSS, ce commentaire malveillant sera publié et stocké tel quel sur le forum. Par conséquent, chaque fois qu'un utilisateur visitera la page contenant ce commentaire, le script JavaScript s'exécutera dans son navigateur.

Ce script peut effectuer toutes sortes d'actions malveillantes, comme voler les cookies de session de l'utilisateur, ce qui pourrait permettre à l'attaquant d'usurper l'identité de l'utilisateur et de prendre le contrôle de son compte.

Méthode Exploitation d'une faille XSS

Pour cet exemple, nous exploitons du code simple en HTML / CSS / PHP afin de montrer comment agit une faille XSS.

En prenant le code suivant php avec SQLite :

```

1 <?php
2 if(isset($_POST['title']) && isset($_POST['message'])) {
3 try {
4     $db = new PDO('sqlite:database.sqlite');
5     $db->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
6
7     $res = $db->exec(
8         "CREATE TABLE IF NOT EXISTS messages (
9             id INTEGER PRIMARY KEY AUTOINCREMENT,
10            title TEXT,
11            message TEXT
12        )"
13    );
14    $stmt = $db->prepare(
15        "INSERT INTO messages (title, message)
16        VALUES (:title, :message)"
17    );
18    $stmt->bindValue(':title', $_POST['title'], SQLITE3_TEXT);
19    $stmt->bindValue(':message', $_POST['message'], SQLITE3_TEXT);
20    $stmt->execute();
21    $messages = $db->query("SELECT * FROM messages");
22    $db = null;
23 } catch (PDOException $ex) {
24     echo $ex->getMessage();
25 }
26 }
27 ?>
28
29 <html>
30 <head>
31     <title>PHP Test</title>
32 </head>
33 <body>
34     <h1>Messages</h1>
35     <form method='post'>
36         <input type="text" placeholder="Titre" name='title' />
37         <input type="text" placeholder="Message" name='message' />
38         <input type="submit" />
39     </form>
40     <?php foreach ($messages as $msg) {

```

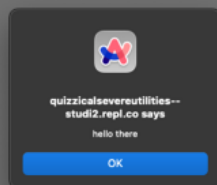
```
41     echo '<p>';
42     echo '<h4>' . $msg['title'] . '</h4>';
43     echo $msg['message'];
44     echo '</p>';
45 } ?>
46 </body>
47 </html>
```

Nous créerons 2 entrées de valeur (input titre et message), non vérifiées qui seront stockées dans la base de données et donc lues par cette dernière lors de la lecture des messages.

Si nous insérons par exemple un script javascript simple comme ce code :

```
1 alert('Hello there')
```

Dans l'input titre ou message, et que nous validons, alors le message Hello There s'affiche à chaque chargement de page car son code est inséré comme un titre ou un message.



Source : capture d'écran personnel du script lancé - Navigateur ARC

Évidemment, ici il ne s'agit que d'un code sans défense qui affiche uniquement un bonjour, mais à partir de cela, il est aussi possible, avec un bon niveau en javascript, de récupérer des données avec l'API javascript et de les envoyer à un serveur pour du vol de donnée, ou de récupérer des scripts JavaScript sur une adresse externe afin de les lancer et se laisser plus de possibilités.

Conseil **Se protéger des failles XSS**

Nous avons pu voir comment une faille XSS pouvait par exemple être exploitée, et allons maintenant voir comment pallier cette problématique.

Pour se protéger contre les attaques XSS avec le code PHP vu précédemment, il serait pertinent de filtrer les entrées utilisateurs et d'échapper tous les caractères spéciaux avant de les inclure dans la page HTML. Ici, les entrées utilisateur sont le titre (`\$_POST['title']`) et le message (`\$_POST['message']`). Nous pourrions donc :

1. Au lieu d'afficher directement `\$_msg['title']` et `\$_msg['message']` dans le HTML, utiliser la fonction `htmlspecialchars()` pour échapper tous les caractères spéciaux.

```
1 echo '<h4>' . htmlspecialchars($_msg['title'], ENT_QUOTES, 'UTF-8') . '</h4>';  
2 echo htmlspecialchars($_msg['message'], ENT_QUOTES, 'UTF-8');
```

La fonction `htmlspecialchars()` permet de convertir les caractères spéciaux en entités HTML. Par exemple, elle convertit `<` en `<`, `>` en `>`, `"` en `"`, et ` ` (apostrophe) en `'`. De cette façon, si du code JavaScript est inclus dans l'input, il sera affiché tel quel et non pas exécuté.

Une autre méthode serait d'utiliser cette fonction avant l'insertion des données.

Avec ces modifications, nos champs seront protégés contre les failles XSS. Cependant, il est primordial de garder à l'esprit que la sécurité des applications est un sujet nécessitant une attention et un apprentissage constants pour se protéger contre divers types d'attaques.

N'hésitez pas à réaliser des tests sur vos sites en injectant du code JavaScript et rappelez-vous qu'il est préférable d'échouer à un test que vous réalisez et perdre du temps à la résorber avant déploiement plutôt qu'à une attaque réelle qui sera possiblement fatale.

C. CSRF

Explication de la faille CSRF

La faille Cross-Site Request Forgery (CSRF), également appelée attaque par contrefaçon de requête intersites, est une vulnérabilité de sécurité dans laquelle un attaquant trompe l'utilisateur pour qu'il exécute des actions non intentionnelles sur une application où il est authentifié.

Illustrons cela, supposons que vous soyez connecté à votre site de banque en ligne dans un onglet et que vous visitiez un autre site dans un autre onglet. Si ce second site est malveillant et conçu pour exploiter une faille CSRF, il pourrait, par exemple, envoyer une requête à votre banque pour effectuer un virement d'argent à partir de votre compte sans que vous vous en rendiez compte. Votre session de banque en ligne étant toujours active et la requête provenant de votre navigateur, la banque la considère comme légitime et ne pourra pas faire de différence.

En pratique, une attaque CSRF peut être réalisée en insérant un script sur un site malveillant qui fera une requête HTTP à l'application cible. Si l'application cible ne met pas en œuvre de mesures de protection CSRF, elle exécutera la requête comme si elle venait de l'utilisateur. Evidemment, les banques sont théoriquement protégées contre ce type d'attaque.

Pour prévenir ces attaques, les développeurs peuvent utiliser des techniques comme l'ajout d'un jeton CSRF dans les formulaires et les requêtes AJAX, qui est ensuite vérifié par le serveur à chaque demande. Les frameworks web modernes ont souvent une protection CSRF intégrée qui peut être activée pour protéger automatiquement les applications.

Cette faille est documentée par l'OWASP à cette adresse (lien en anglais) : <https://owasp.org/www-community/attacks/csrf>.

Fonctionnement de la faille CSRF

Reprenons l'exemple précédent afin d'expliquer le fonctionnement, supposons que vous soyez connecté à votre compte bancaire qui a une faille CSRF (cas extrêmement rare, mais possible en raison de l'externalisation de certains développements).

Pendant que vous êtes connecté, vous recevez un e-mail vous incitant à cliquer sur un lien vers un site innocent en apparence (sauvons les bébés baleines par exemple). Ce site Web peut être contrôlé par un attaquant qui cherche à exploiter cette faille CSRF.

Sur le site Web, une image est censée être affichée, mais le lien source de l'image est en réalité une requête HTTP conçue pour transférer des fonds de votre compte bancaire à celui de l'attaquant. La requête pourrait alors ressembler à <http://mabanquemaalprotegee.fr/transfer?account=mechant&amount=1000&from=gentil>.

Lorsque votre navigateur tente de charger l'image, il envoie automatiquement la requête vers votre banque.

Comme vous êtes toujours connecté à celle-ci, cette requête est associée à votre session active (votre compte actuel). Si votre banque ne fait pas de vérification CSRF, elle peut croire que cette requête est une action légitime de votre part et procéder au transfert d'argent sans poser plus de question.

En ajoutant simplement une image avec une telle URL sur une page Web, un attaquant pourrait alors effectuer une action en votre nom sans que vous n'en ayez conscience. Ici nous avons parlé d'un site bancaire mais cela peut avoir lieu dans un contexte professionnel, par exemple avec le site de votre entreprise, afin de récupérer rapidement les données des clients.

Méthode Exploitation d'une faille CSRF

Si nous reprenons notre code précédent en enregistrant les messages avec la méthode GET et non POST, nous pourrions avoir la faille CSRF suivante :

```

1 <html>
2   <head>
3     <title>CSRF Test</title>
4   </head>
5   <body>
6     <h1>CSRF</h1>
7     
8     <p>Cette page charge une image afin d'envoyer une requête au site précédent avec le même
9     identifiant et ip</p>
10  </body>
11 </html>
```

Comme vous pouvez le constater, il s'agit simplement de HTML. Si la faille existe, il est donc très simple de l'exploiter.

Dans le cas actuel, cela ne rajoute qu'un message à notre liste, et hormis l'ip il n'y a pas de réel authentification, mais cela peut s'aggraver très facilement, notamment sur la base de cet exemple, publiez un article de phishing à votre place sur le site de votre entreprise.

Conseil Se protéger des failles CSRF

Afin de pouvoir éviter cette vulnérabilité, Il existe de multiple méthode, que ce soit en tant que développeur mais aussi en tant qu'utilisateur, notamment :

En tant que développeur, nous pouvons trouver plusieurs mesures à mettre en place pour protéger nos application contre ces attaques :

1. L'utilisation de jeton CSRF : lorsqu'un utilisateur se connecte à notre application, nous générons un jeton CSRF unique et l'associons à sa session. Ce jeton doit être inclus dans chaque formulaire de l'application et chaque requête AJAX. Lorsque le serveur reçoit une requête, il vérifie alors que le jeton CSRF associé à la session correspond au jeton CSRF de la requête.
2. L'utilisation du principe SameSite pour les cookies : si notre application utilise des cookies pour gérer les sessions, nous pouvons définir l'attribut SameSite du cookie. Cela indique au navigateur de n'envoyer le cookie que si la requête provient du même site que celui qui a établi le cookie. Cela peut entraver les attaques

CSRF car le navigateur n'inclura pas le cookie de session lorsqu'un utilisateur suit un lien depuis un autre site que notre application. (dans ce domaine, Google Chrome inclut automatiquement de bonne sécurité depuis 2021).

3. La vérification de l'origine ou le referer des requêtes : notre serveur peut vérifier l'en-tête d'origine ou le referer des requêtes pour s'assurer qu'elles proviennent de notre site et non d'un site externe, possiblement malveillant.

De l'autre côté, en tant qu'utilisateur, quelques mesures peuvent aussi être prise pour se protéger contre les attaques CSRF :

1. Se déconnecter des applications sensibles directement et manuellement en fin d'utilisation. Cela ferme la session et rend les attaques CSRF avec les identifiants impossibles.
2. Faire attention aux sites non fiables ou suspects, dans le cas de l'utilisation d'un de cela, prioriser une solution comme la navigation privée.
3. Être prudent en cliquant sur des liens dans des e-mails ou des messages. Ces derniers peuvent rediriger vers un site malveillant qui tente une attaque CSRF.
4. Utilisez un logiciel antivirus et anti-malware qui comprend une protection contre ces attaques.

Il existe donc, de nombreuses méthodes pour se protéger contre ces failles, en tant qu'utilisateur comme développeur.

Complément Les frameworks se protégeant des failles CSRF

Pour un développeur, une solution fiable est aussi d'utiliser une bibliothèque ou un framework intégrant nativement une protection contre les failles CSRF, on peut par exemple citer en PHP, le framework Symfony qui inclut des jetons CSRF grâce à sa bibliothèque « *symfony/security-csrf* ».

Méthode

D. Exercice : Quiz

[solution n°1 p.21]

Question 1

Qu'est-ce qu'une faille de sécurité dans le développement ?

- ☐ C'est une fonctionnalité qui fait planter le logiciel
- ☐ C'est un défaut qui permet aux attaquants d'accéder ou de compromettre les informations
- ☐ C'est un luxe que seules les grandes entreprises peuvent se permettre
- ☐ C'est une fonctionnalité supplémentaire qui améliore les performances du logiciel

Question 2

Qu'est-ce qu'une attaque XSS (Cross-Site Scripting) ?

- ☐ C'est une technique de conception graphique
- ☐ C'est l'utilisation de scripts sur un site web pour transférer des informations à un autre site
- ☐ C'est l'exploitation d'une faille de sécurité où un attaquant injecte du code malveillant dans une page web
- ☐ C'est une méthode pour stocker des informations à l'aide de cookies

Question 3

Comment un développeur peut-il se protéger contre les failles XSS ?

- ☐ En évitant l'utilisation de JavaScript
- ☐ En entrant lui-même les informations des utilisateurs
- ☐ En utilisant des techniques de validation des entrées, et en utilisant des politiques de sécurité de contenu
- ☐ En n'utilisant que des technologies backend

Question 4

Qu'est-ce qu'une attaque CSRF (Cross-Site Request Forgery) ?

- ☐ C'est une attaque qui force un utilisateur authentifié à exécuter une action non intentionnelle
- ☐ C'est une attaque qui vise à voler les mots de passe des utilisateurs
- ☐ C'est une attaque qui empêche les utilisateurs d'accéder à un site web
- ☐ C'est une attaque qui vise à surcharger un serveur avec du trafic

Question 5

Comment un développeur peut-il se protéger contre les attaques CSRF ?

- ☐ En ajoutant des captchas à tous les formulaires
- ☐ En générant et vérifiant des jetons CSRF à chaque demande
- ☐ En interdisant l'utilisation des cookies
- ☐ En utilisant uniquement des méthodes de requête GET

III. Autres failles de sécurité

A. Les failles de Base de donnée

Explication des différentes failles de base de donnée (accès, injection)

Les failles de sécurité au sein des bases de données représentent une des menaces majeures pour la confidentialité et l'intégrité des données. Parmi les plus courantes, on trouve les failles d'accès et d'injection.

- Les failles d'accès se produisent quand les utilisateurs non autorisés obtiennent un accès à des ressources de base de données sensibles ou un utilisateur accède à des fonctionnalités auquel il ne devrait pas avoir accès. Cela peut se produire si les politiques de contrôle d'accès ne sont pas correctement mises en place ou si les informations d'identification à la base de données (comme les noms d'utilisateur et les mots de passe) sont compromises.

Par exemple, si un attaquant volait les informations d'identification à la base de données d'un administrateur root, il pourrait avoir accès à toutes les données qu'elle contient. Pour se prémunir contre de telles failles, il est crucial de mettre en place des politiques de contrôle d'accès robustes, de chiffrer les informations d'identification, de créer une politique de droit limités et d'utiliser des mots de passe forts et uniques.

- Les failles d'injection, quant à elles, se produisent lorsque des attaquants insèrent ou « injectent » des instructions de code dans une requête de base de données, souvent cela passe par des données d'entrée qui ne sont pas correctement validées.

La plus connue de ces attaques est l'injection SQL, où un attaquant manipule les instructions SQL pour obtenir des informations non autorisées, modifier les données, ou même exécuter des commandes arbitraires sur la base de données.

Pour se prémunir contre les injections, il est important de valider correctement toutes les données d'entrée, et d'utiliser des requêtes préparées ou des procédures stockées pour séparer les données des instructions de code.

Les failles de base de données sont aujourd'hui l'une des sources de failles les plus accessibles pour un attaquant car il existe des scripts pré-fait et l'une des plus grave, car elle permet de toucher directement les données.

Explication des problèmes des droits d'accès

La gestion des droits d'accès est un point critique de la sécurité des bases de données. Elle détermine qui peut accéder à la base de données et ce qu'ils sont autorisés à faire avec les données qu'elle contient. Les erreurs de gestion des droits peuvent entraîner des fuites de données, des modifications non autorisées et d'autres types d'abus plus ou moins destructeurs.

L'une des problématiques, concernant les droits d'accès, est l'accès qu'elle donne à la base de données. Les informations d'identification permettant d'accéder à la base de données ne doivent jamais être divulguées ou conservées dans des endroits non sécurisés. Par exemple, un certain nombre de failles ont été causées par des informations d'identification de base de données stockées en clair dans un code source ou transmises sans être chiffrées.

Aussi, le principe du moindre privilège doit être utilisé lors de l'attribution des droits d'accès. Cela signifie que les utilisateurs et les applications ne doivent avoir que le minimum de droits nécessaires pour accomplir leurs tâches. Par exemple, un utilisateur qui a seulement besoin de lire des données pour générer des rapports n'a pas besoin d'avoir des droits d'écriture. Au-delà de l'aspect failles de sécurité, ce principe permet d'éviter les erreurs humaines potentiellement destructives. Une analogie serait que vous n'iriez pas donner vos clés à une personne qui vous rend visite.

Il est également important de surveiller et contrôler l'accès aux données sensibles. Par exemple, si une base de données contient des informations de paiement des clients, seulement quelques rôles privilégiés devraient avoir l'accès à ces données et ces dernières ne devraient pas être stockées en clair.

Cette gestion des droits d'accès doit être dynamique et évoluer avec les changements de rôles et de responsabilités. Lorsqu'un employé quitte une entreprise ou change de poste, ses anciens droits d'accès doivent être révoqués immédiatement pour éviter tout abus potentiel ou création de faille.

Pour finir, il est important que les accès à la base de données ne soient possibles qu'en interne depuis l'entreprise. Ainsi il sera nécessaire d'accéder au réseau de l'entreprise ou au VPN de cette dernière afin de pouvoir tenter une attaque.

Explication des injections SQL et No-SQL

Une injection SQL est une faille de sécurité courante où un attaquant insère du code SQL malveillant à travers les entrées utilisateur d'une application. Par exemple, à partir d'un formulaire de connexion, un attaquant pourrait entrer un code SQL qui modifie la requête, ce qui pourrait permettre à ce dernier de contourner l'authentification sans avoir à connaître les informations d'identification, ou encore de supprimer des tables de la base de données.

Pour prévenir les injections SQL, il est essentiel d'échapper correctement toutes les entrées utilisateur et de toujours utiliser des requêtes préparées ou des procédures stockées.

Une injection NoSQL, quant à elle, est similaire à une injection SQL, mais elle cible les bases de données NoSQL. Comme ces bases de données n'utilisent pas le langage SQL, ces attaques se font souvent par l'insertion de scripts malveillants ou l'envoi de requêtes mal formées. Par exemple, un attaquant pourrait tenter d'exploiter une faille en envoyant une requête qui modifie la logique de l'opération de la base de données. Là encore, l'échappement et la validation appropriés des entrées utilisateur sont un élément clé de la prévention des injections NoSQL.

Dans ces deux cas, comprendre la structure et le fonctionnement des requêtes des bases de données est essentiel pour identifier et prévenir ce type d'attaque. Il est également crucial de mettre à jour régulièrement les bases de données et leurs pilotes, afin de bénéficier des derniers correctifs de sécurité.

Définition Injections SQL

Une attaque par injection SQL consiste en l'insertion ou l'injection d'une requête SQL via les données d'entrée provenant du client vers l'application. Une exploitation réussie de l'injection SQL peut lire des données sensibles de la base de données, modifier des données de la base de données (Insérer/Mettre à jour/Supprimer), exécuter des opérations d'administration sur la base de données (comme l'arrêt du SGDB), récupérer le contenu d'un fichier donné présent sur le système de fichiers du SGDB et dans certains cas, émettre des commandes à l'exploitation système.

La fiche OWAST de cette faille est disponible à cette adresse : https://owasp.org/www-community/attacks/SQL_Injection.

Méthode Réalisation d'une injection SQL

Pour illustrer une injection SQL, prenons l'exemple d'un formulaire de connexion classique qui demande un nom d'utilisateur et un mot de passe.

Normalement, ces informations sont utilisées pour construire une requête SQL qui vérifie si l'utilisateur existe dans la base de données. Dans une situation normale, la requête créée par le développeur pourrait ressembler à :

```
1 SELECT * FROM Utilisateurs WHERE Nom = 'nom_utilisateur' AND MotDePasse = 'mot_de_passe';
```

Cependant, un attaquant peut essayer de manipuler cette requête.

Par exemple, pour le nom d'utilisateur, l'attaquant peut entrer : ' OR '1'='1. Ceci modifie la requête SQL en :

```
1 SELECT * FROM Utilisateurs WHERE Nom = '' OR '1'='1' AND MotDePasse = 'mot_de_passe';
```

Étant donné que '1'='1' est toujours vrai, cette requête retourne tous les utilisateurs, permettant à l'attaquant de se connecter sans connaître le véritable nom d'utilisateur ou mot de passe.

Un autre exemple plus puissant, imaginons que ce même site web dispose d'un formulaire qui permet aux utilisateurs de mettre à jour leur adresse email.

Dans une situation normale, une requête à partir de ce formulaire pourrait ressembler à :

```
1 UPDATE Utilisateurs SET Email = 'mymail@mailier.fr WHERE Nom = 'monSuperNom';
```

Cependant, l'attaquant pourrait chercher à manipuler cette requête pour changer l'adresse email de tous les utilisateurs. Il pourrait entrer dans le champ de l'adresse email la valeur suivante : dummy@email.com'; UPDATE Utilisateurs SET Email = 'attaquant@exemple.com' --

La requête SQL devient alors :

```
1 UPDATE Utilisateurs SET Email = 'mymail@mailier.fr'; UPDATE Utilisateurs SET Email = 'myBatmail@mailier.fr' -- WHERE Nom = 'monSuperNom';
```

Dans un premier temps la requête change l'adresse email de l'utilisateur(monSuperNom) comme prévu en mymail@mailier.fr, mais dans un second temps, elle modifie aussi l'adresse email de tous les utilisateurs afin qu'elle devienne myBatmail@mailier.fr.

Gardez à l'esprit qu'il est interdit aux développeurs de tester les injections SQL sur des sites qui ne sont pas les leurs, sans permission explicite (et par conseil, écrite).

En plus d'être illégal, ces dernières peuvent causer des dommages potentiellement irréparables à la base de données du site et sont traçables facilement en général.

Méthode

B. Autres failles

Présentation d'autres failles de sécurité (inclusion de fichiers et l'authentification insuffisante, clé mal sécurisée)

En dehors des injections SQL, des failles XSS et CSRF, il existe d'autres types de failles de sécurité auxquelles les développeurs sont confrontés, on peut notamment citer :

- L'inclusion de fichiers : cette faille consiste en un attaquant qui inclut un fichier externe dans une application. En exploitant cette faille, un attaquant pourrait exécuter du code malveillant, accéder à des informations sensibles, voire prendre le contrôle total de l'application.
- L'authentification insuffisante : une faille provoquée lorsque les mécanismes d'authentification ne sont pas suffisamment robustes, permettant ainsi à un attaquant de contourner les contrôles d'accès. Si un attaquant devine ou obtient les informations d'identification d'un utilisateur (comme son nom d'utilisateur et son mot de passe, par exemple), il pourra accéder à l'application en tant que cet utilisateur et obtenir un accès non autorisé à des informations sensibles.
- Clés mal sécurisées : dans les applications où l'authentification est basée sur des clés, comme les tokens JWT (Jetons Web JSON), si ces clés ne sont pas correctement configurées et sécurisées, elles peuvent être volées ou compromises. Un attaquant peut alors utiliser cette clé pour accéder à l'application, souvent avec des privilèges élevés.

Chacune de ces failles peuvent avoir des conséquences graves directes si elles sont exploitées mais aussi indirectes en permettant de créer une porte d'entrée afin de monter en privilèges et attaquer une fois le niveau d'accès maximum atteint.

Il est donc essentiel pour les développeurs de comprendre ces vulnérabilités et de mettre en œuvre des mesures pour se protéger contre elles, telles que le filtrage et la validation des entrées utilisateur, l'utilisation de mécanismes d'authentification robustes et le stockage sécurisé des clés et des informations d'identification. Ces dernières restent un échantillon léger des failles existantes.

Conseil Utilisation du 2FA

Le 2FA, ou l'authentification à deux facteurs, est une option de sécurité qui requiert deux types de preuves d'identité afin qu'un utilisateur puisse accéder à son compte. C'est une couche de sécurité supplémentaire conçue afin de garantir qu'un utilisateur est bien celui qu'il prétend être et ainsi protéger les comptes contre tout accès non autorisé.

L'authentification à deux facteurs fonctionne, le plus souvent, en combinant quelque chose que l'utilisateur sait (comme un mot de passe) avec quelque chose que l'utilisateur a (comme un téléphone sur lequel il peut recevoir un code de validation) ou quelque chose qui est propre à l'utilisateur (comme son empreinte digitale).

Par exemple, après avoir entré leur mot de passe, les utilisateurs peuvent être invités à entrer un code envoyé à leur téléphone par message (comme Google), ou à valider une notification envoyée à une application d'authentification sur leur téléphone (comme Blizzard).

Le 2FA est donc une méthode très efficace pour augmenter la sécurité des comptes utilisateurs, en rendant plus difficile pour un attaquant de gagner l'accès à un compte, sans pour autant compliquer de façon critique l'expérience utilisateur.

Beaucoup de services en ligne importants, comme les comptes de messagerie, les comptes bancaires et les réseaux sociaux, offrent aujourd'hui la possibilité d'activer l'authentification à deux facteurs. Il est donc fortement recommandé aux utilisateurs de profiter de cette fonctionnalité pour augmenter la sécurité de leurs comptes.

Exemple Microsoft et la sécurité en 3 étapes

Microsoft offre une sécurité intéressante en termes de 2FA par message SMS, cette dernière est en trois étapes pour protéger les comptes des utilisateurs, ce qui offre une couche de protection supplémentaire au-delà de la simple utilisation d'un mot de passe et évite les dérives en cas de mot de passe corrompu.

Identification : la première étape consiste en un mot de passe unique qu'un utilisateur doit créer lors de l'inscription d'un compte Microsoft.

Microsoft recommande alors fortement l'utilisation d'un mot de passe fort, qui est un mélange de lettres, chiffres, et de caractères spéciaux pour rendre la tâche difficile à quelqu'un essayant de deviner ou de cracker le mot de passe.

Authentification : la deuxième étape vérifie les quatre derniers chiffres du numéro de téléphone enregistré sur le compte. Après avoir entré correctement le mot de passe, Microsoft demandera à l'utilisateur de confirmer ces quatre chiffres. C'est une méthode de vérification qui assure que l'utilisateur possède effectivement le numéro de téléphone associé à l'adresse e-mail du compte. Dans cette optique, un attaquant doit connaître le mot de passe et le numéro de téléphone associé.

Autorisation : la troisième et dernière étape est l'envoi d'un code de vérification par message au numéro de téléphone associé au compte. L'utilisateur doit entrer ce code sur le site pour prouver qu'il a réellement accès au téléphone. C'est une forme d'authentification à deux facteurs, basique, qui offre une sécurité supplémentaire.

En combinant ces trois étapes - le mot de passe, la vérification des quatre derniers chiffres du numéro de téléphone et le code de vérification envoyé par message - Microsoft fournit une méthode plus robuste du 2FA mais aussi plus confortable pour l'utilisateur.

Ce dernier connaît effectivement son numéro de téléphone en général (90 % d'un panel le connaissent selon le parisien en 2021), cette étape intermédiaire n'est donc pas problématique et évite, dans le cas d'un attaquant passant le mot de passe, de favoriser la faille de sécurité humaine pour la 3^{ème} étape.

Complément La failles de sécurité humaine

La faille de sécurité humaine, souvent considérée comme le maillon le plus faible de la chaîne de sécurité, décrit les erreurs ou les négligences commises par les individus qui peuvent conduire à des brèches de sécurité.

Même avec les systèmes de sécurité les plus robustes en place, les actions imprudentes ou mal informées d'une personne peuvent rendre une organisation vulnérable aux attaques.

Un exemple courant de cette faille est l'utilisation de mots de passe faibles, simples ou répétitifs. Malgré les fréquentes recommandations de sécurité sur l'importance d'utiliser des mots de passe forts et uniques, beaucoup de personnes continuent d'utiliser des mots de passe prévisibles ou réutilisent le même mot de passe sur plusieurs comptes, rendant plus facile pour les attaquants d'accéder à leurs informations sensibles.

En parallèle, le phishing est une autre méthode d'attaque qui exploite la faille de sécurité humaine. Les attaquants envoient des e-mails apparemment légitimes dans le but de tromper les individus afin qu'ils révèlent des informations sensibles, comme des identifiants de connexion ou les numéros de carte de crédit.

Cela peut aussi être pour des mots de passe ou code de sécurité 2FA.

La formation et la sensibilisation sont les moyens les plus efficaces pour combler cette faille.

En formant les individus à reconnaître et à éviter les pratiques de sécurité dangereuses, les organisations peuvent renforcer considérablement leur sécurité mais aussi l'intérêt de leur employé à cet aspect.

C. Veilles et métier

Présentation rapide de la veille technologique sur les failles et du métier de pentester

La veille technologique s'effectue sur énormément de sujets mais doit aussi se centrer sur les failles de sécurité, celle-ci joue un rôle clé dans le renforcement de la sécurité des applications.

Compte tenu de l'évolution rapide des technologies et de l'ingéniosité croissante des attaquants, assisté de plus en plus par l'IA, il est essentiel de rester informé des dernières vulnérabilités découvertes, des nouvelles techniques d'attaque et des évolutions des normes de sécurité. Cela permet ainsi de combler proactivement les lacunes potentielles dans les défenses et d'amortir les impacts des attaques potentielles.

Pour cela, il existe un métier spécialisé dans ce domaine, le métier de pentester, ou testeur d'intrusion, dont le rôle est l'un des plus critiques.

Les pentesters sont des professionnels de la sécurité qui sont chargés de trouver et d'exploiter les vulnérabilités des applications, dans le but d'identifier les failles de sécurité avant qu'elles ne puissent être exploitées par des personnes malveillantes.

Ils simulent des attaques sur les réseaux, les applications, les appareils et d'autres points d'accès potentiel pour évaluer la robustesse des systèmes de sécurité et recommander des améliorations. Les pentesters ont donc un rôle essentiel pour aider les organisations à anticiper et à prévenir activement les failles de sécurité.

Méthode	Veille technologique sur les failles
---------	--------------------------------------

Assurer une veille technologique efficace sur les failles de sécurité est primordial afin de garder un standard qualitatif en termes de sécurité.

Tout d'abord, l'étape de base est de s'abonner à des sources d'information réputées qui publient régulièrement des bulletins sur les dernières vulnérabilités découvertes.

Certains sites ou organismes comme CVE (Common Vulnerabilities and Exposures) ou l'ANSSI (Agence Nationale de la Sécurité des Systèmes d'Information) plus spécifique à la France, publient régulièrement des bulletins détaillés sur les failles découvertes au fur et à mesure.

Il est également important de maintenir une liste actualisée des technologies et des outils utilisés au sein des divers projets. Cela permet de cibler la veille technologique et de se concentrer sur les vulnérabilités qui sont pertinentes pour l'environnement utilisé.

Ensuite, il est nécessaire de prendre un temps afin d'examiner régulièrement les nouvelles vulnérabilités rapportées et d'évaluer leur gravité.

Pour finir, une fois qu'une faille potentiellement dangereuse est identifiée, il convient d'élaborer et mettre en œuvre un plan d'action pour y remédier. Cela peut impliquer la mise à jour ou le patching des outils ou application, la modification des configurations de sécurité, ou dans certains cas, le remplacement complet de la partie vulnérable.

La veille technologique concernant les failles de sécurité est donc une activité continue qui doit être intégrée dans la stratégie de sécurité globale et la stratégie de veille technologique du développeur afin d'assurer une défense optimale contre les menaces émergentes.

Un parallèle peut alors être fait avec l'aviation, ou certain organisme étudie les crash afin de favoriser la connaissance et la sécurité commune. Il est primordial de ne pas ignorer ces derniers. Il en est de même pour les failles de sécurité.

Complément	Importance de prendre du recul sur les failles présentes
------------	--

En tant que développeur, prendre du recul et examiner régulièrement son propre code à la recherche des failles de sécurité potentielles ainsi que tester soit même les failles les plus récurrentes est une étape essentielle dans le processus de développement.

Parfois, lorsqu'on est absorbé dans l'écriture du code et la résolution des problèmes fonctionnels, il devient facile de négliger certains aspects comme la sécurité. Cependant, laisser ces failles de sécurité dans le code peut rendre l'application vulnérable à des attaques potentielles et donc engendrer de lourdes conséquences.

Prendre le temps de revoir son code à la recherche de faiblesses au niveau de la sécurité permet de corriger ces problèmes avant que le code ne soit mis en production.

Cela peut inclure la recherche de problèmes tels que la non-validation des entrées utilisateur, l'utilisation de composants avec des vulnérabilités connues, ou l'exposition accidentelle de données confidentielles (l'envoi de clé API sur github par exemple).

Mais aussi, l'examen régulier de son code offre une excellente occasion de croissance et d'apprentissage. Il permet d'identifier et de comprendre les erreurs ou les mauvaises pratiques de codage courantes qui pourraient conduire à des vulnérabilités de sécurité, et d'acquérir des compétences pour améliorer la qualité du code à l'avenir.

Enfin, il est aussi possible d'utiliser des outils d'automatisation de test de vulnérabilité, ou de qualité du code (à l'image de SonarQube) afin d'avoir un retour objectif sur ces derniers.

Attention cependant, cela ne vaut pas une recherche et test manuel, ce sont des outils facilitant la détection uniquement.

D. Exercice : Quiz

[solution n°2 p.22]

Question 1

Quelles sont les deux principales vulnérabilités reconnues dans le domaine des bases de données ?

- ☐ Les failles d'accès
- ☐ Les failles d'injection
- ☐ Les chevaux de Troie
- ☐ Les Spam

Question 2

Quelle est la définition d'une faille d'injection SQL ?

- ☐ Une méthode pour injecter du code
- ☐ Une technique pour contourner l'authentification
- ☐ Un moyen de pirater les bases de données
- ☐ Un type de virus informatique

Question 3

Quelles sont les trois étapes de la méthode de sécurité à deux facteurs (2FA) de Microsoft ?

- ☐ Authentification, validation, et vérification
- ☐ Entrée, confirmation, et approbation
- ☐ Identification, authentification et autorisation
- ☐ Authentification, authentification, et authentification

Question 4

Quelle fonction un testeur d'intrusion (pentester) remplit-il ?

- ☐ Développer des logiciels
- ☐ Tester les vulnérabilités de la sécurité
- ☐ Concevoir des sites Web
- ☐ Rédiger des rapports techniques

Question 5

Quelle est l'importance de la surveillance des vulnérabilités de sécurité ?

- ☐ Se tenir au courant des dernières menaces
- ☐ Se tenir au courant des derniers logiciels
- ☐ Pour maintenir les performances du système
- ☐ Pour améliorer la productivité du travail

IV. L'essentiel

Les failles de sécurité sont donc un enjeu majeur dans le domaine des technologies et plus particulièrement sur la création d'application, de façon encore plus importante lors de l'utilisation de bases de données.

Il existe de nombreuses failles de sécurité, notamment les failles XSS et CSRF qui seront axées sur une utilisation côté client mais aussi des failles de base de données par exemple.

Dans ce domaine les deux vulnérabilités les plus courantes sont les failles d'accès et les failles d'injection SQL.

Les failles d'accès concernent les autorisations d'accéder à des données ou à des ressources, rendant possible le visionnage, la modification ou la suppression non autorisés de données.

Les failles d'injection SQL, quant à elles, permettent à un attaquant d'insérer du code SQL dans une requête, compromettant ainsi l'intégrité et la sécurité des données.

Enfin, il existe aussi les failles de sécurité d'authentification et humaine.

Pour contrer ces dernières, des mesures de sécurité telles que l'authentification à deux facteurs (2FA) peuvent être mises en place, comprenant des étapes d'identification, parfois d'authentification et d'autorisation.

De plus, il existe des testeurs d'intrusion, également connus sous le nom de pentesters, jouent un rôle crucial en testant les vulnérabilités dans des systèmes et applications, permettant d'anticiper les attaques.

Pour finir, il reste primordial d'effectuer une surveillance des vulnérabilités pour rester à jour avec les dernières menaces et maintenir la sécurité des applications.

V. Auto-évaluation

A. Exercice

Au sein d'une entreprise de conception de site, un client Ayden Should Hours se présente afin de vous demander un site e-commerce, votre responsable de projet vous fournit alors la première page du cahier des charges de ce futur site.

Question 1

[solution n°3 p.23]

À partir du cahier des charges suivants, établissez quel failles de sécurité seront à surveiller (pour cette partie, seul les failles de sécurité évoqués seront demandée).

Cahier des charges site Ayden Should Hours.

I. Introduction

L'objectif de ce cahier des charges est de définir les exigences et les attentes en termes de conception, et de fonctionnalité pour le développement d'un site de commerce électronique pour la marque de shampoing « *Ayden Should Hours* ». Le site doit offrir une expérience utilisateur fluide et séduisante, conçue pour faciliter l'achat de shampoings et autres produits capillaires de haute qualité sous la marque « *Ayden Should Hours* ». Tout en assurant une sécurité afin de préserver l'image de la marque, Il doit en outre être optimisé pour le référencement afin d'assurer une visibilité maximale sur les moteurs de recherche.

II. Spécifications détaillées

1. Utilisation de formulaire :

Nous utilisons des formulaires en méthode GET afin de pouvoir ajouter les articles au paniers.

2. Connexion :

Nous réaliserons une connexion permanente afin de connecter une seule fois le client.

3. Base de donnée SQL :

Nous utilisons des requêtes simples, rapide pour communiquer avec la base de données SQL.

4. Rapidité :

Il sera essentiel de déployer rapidement le site, seul quelques tests visuels des pages devront être réalisés.

Question 2

[solution n°4 p.23]

À partir de la liste des failles, définissez quel type d'action serait utile concernant les failles de sécurité de base de donnée SQL.

B. Test

Exercice 1 : Quiz

[solution n°5 p.23]

Question 1

Qu'est-ce qu'une faille de sécurité ?

- ☐ Un défaut dans une théorie ou un argument
- ☐ Une erreur dans un système de sécurité
- ☐ Un logiciel obsolète
- ☐ Toutes ces réponses sont incorrectes

Question 2

Quel est le risque d'avoir des vulnérabilités dans une application ?

- ☐ Permettre des attaques automatisées
- ☐ Permettre à des logiciels malveillants et à des programmes non autorisés de fonctionner
- ☐ Il n'y a pas de risques
- ☐ Toutes ces réponses sont incorrectes

Question 3

Comment se protéger efficacement contre les cyber-attaques ?

- ☐ En ignorant le problème
- ☐ En activant le 2FA
- ☐ En utilisant le même mot de passe pour tous vos comptes
- ☐ En créant les applications avec le concept de security by design

Question 4

Que signifie XSS ?

- ☐ Un type de cookie
- ☐ Cross-Site Scripting
- ☐ Le nouveau nom de twitter
- ☐ Xylophone Salulaire des Salamandres

Question 5

Qu'est-ce que la faille Cross-Site Request Forgery (CSRF) ?

- ☐ Une vulnérabilité qui permet d'injecter du code
- ☐ Une technique d'hameçonnage qui cible les e-mails des utilisateurs
- ☐ Une vulnérabilité qui force un utilisateur à exécuter des actions non désirées
- ☐ Toutes ces réponses sont incorrectes

Question 6

Qu'est-ce qu'une faille d'accès dans une base de données ?

- ☐ Une faille qui permet de modifier les données sans autorisation
- ☐ Une faille qui permet à des attaquants non autorisés d'obtenir un accès à la base de données
- ☐ Une faille qui permet à un utilisateur d'accéder à des fonctionnalités auxquelles il ne devrait pas avoir accès
- ☐ Toutes ces réponses sont incorrectes

Question 7

À quoi sert l'authentification à deux facteurs (2FA) ?

- ☐ À augmenter la sécurité des connexions aux comptes
- ☐ À simplifier le processus de connexion
- ☐ À récupérer les mots de passe oubliés plus facilement
- ☐ Toutes ces réponses sont incorrectes

Question 8


Qu'est-ce que l'injection SQL ?

- ☐ Une technique d'attaque qui insère du code SQL
- ☐ Une technique de développement qui permet d'améliorer l'efficacité des requêtes SQL
- ☐ Une erreur de programmation courante lors de l'écriture des requêtes SQL
- ☐ Toutes ces réponses sont incorrectes

Solutions des exercices


Exercice p. 9 Solution n°1**Question 1**

Qu'est-ce qu'une faille de sécurité dans le développement ?

- ☐ C'est une fonctionnalité qui fait planter le logiciel
- ☒ C'est un défaut qui permet aux attaquants d'accéder ou de compromettre les informations
- ☐ C'est un luxe que seules les grandes entreprises peuvent se permettre
- ☐ C'est une fonctionnalité supplémentaire qui améliore les performances du logiciel
-  Une faille de sécurité, aussi appelée vulnérabilité de sécurité, est une faiblesse ou un défaut qui peut être exploité par un attaquant pour accéder ou compromettre de manière non autorisée aux informations.


Question 2

Qu'est-ce qu'une attaque XSS (Cross-Site Scripting) ?

- ☐ C'est une technique de conception graphique
- ☐ C'est l'utilisation de scripts sur un site web pour transférer des informations à un autre site
- ☒ C'est l'exploitation d'une faille de sécurité où un attaquant injecte du code malveillant dans une page web
- ☐ C'est une méthode pour stocker des informations à l'aide de cookies
-  Une attaque XSS est une vulnérabilité de sécurité où un attaquant injecte du code malveillant dans une page web, qui est ensuite exécuté par l'utilisateur final.


Question 3

Comment un développeur peut-il se protéger contre les failles XSS ?

- ☐ En évitant l'utilisation de JavaScript
- ☐ En entrant lui-même les informations des utilisateurs
- ☒ En utilisant des techniques de validation des entrées, et en utilisant des politiques de sécurité de contenu
- ☐ En n'utilisant que des technologies backend
-  Les développeurs peuvent se prémunir contre les failles XSS en utilisant diverses techniques, notamment en validant et échappant correctement les entrées utilisateur, et en mettant en œuvre des politiques de sécurité de contenu pour contrôler ce qui peut être chargé sur une page.

Question 4


Qu'est-ce qu'une attaque CSRF (Cross-Site Request Forgery) ?

- ☒ C'est une attaque qui force un utilisateur authentifié à exécuter une action non intentionnelle
- ☐ C'est une attaque qui vise à voler les mots de passe des utilisateurs
- ☐ C'est une attaque qui empêche les utilisateurs d'accéder à un site web
- ☐ C'est une attaque qui vise à surcharger un serveur avec du trafic
-  Une attaque CSRF est une attaque qui trompe l'utilisateur pour qu'il exécute des actions non intentionnelles sur une application web où il est authentifié.

Question 5

Comment un développeur peut-il se protéger contre les attaques CSRF ?

- ☐ En ajoutant des captchas à tous les formulaires
- ☒ En générant et vérifiant des jetons CSRF à chaque demande
- ☐ En interdisant l'utilisation des cookies
- ☐ En utilisant uniquement des méthodes de requête GET


 Les développeurs peuvent se protéger contre les attaques CSRF en générant un jeton CSRF unique lorsqu'un utilisateur se connecte, et en l'incluant pour toutes les demandes sensibles. Lorsque le serveur reçoit la demande, il vérifie que le jeton CSRF correspond à la session de l'utilisateur, assurant donc que la demande est légitime.

Exercice p. 16 Solution n°2

Question 1

Quelles sont les deux principales vulnérabilités reconnues dans le domaine des bases de données ?


- ☒ Les failles d'accès
- ☒ Les failles d'injection
- ☐ Les chevaux de Troie
- ☐ Les Spam

 Les deux principales vulnérabilités signalées dans les bases de données sont les failles d'accès et les failles d'injection SQL.

Question 2

Quelle est la définition d'une faille d'injection SQL ?

- ☒ Une méthode pour injecter du code
- ☐ Une technique pour contourner l'authentification
- ☒ Un moyen de pirater les bases de données
- ☐ Un type de virus informatique

 Une faille d'injection SQL est une technique d'attaque où un attaquant peut insérer du code malveillant SQL dans une requête afin d'accéder à la base de données directement ou indirectement.

Question 3

Quelles sont les trois étapes de la méthode de sécurité à deux facteurs (2FA) de Microsoft ?

- ☐ Authentification, validation, et vérification
- ☐ Entrée, confirmation, et approbation
- ☒ Identification, authentification et autorisation
- ☐ Authentification, authentification, et authentification

- Q L'authentification à deux facteurs (2FA) de Microsoft fonctionne en trois étapes : identification, authentification et autorisation.

Question 4

Quelle fonction un testeur d'intrusion (pentester) remplit-il ?

- ☐ Développer des logiciels
- ☒ Tester les vulnérabilités de la sécurité
- ☐ Concevoir des sites Web
- ☐ Rédiger des rapports techniques

- Q Un pentester, ou un testeur d'intrusion, est généralement employé pour tester les vulnérabilités dans les systèmes de sécurité.

Question 5

Quelle est l'importance de la surveillance des vulnérabilités de sécurité ?

- ☒ Se tenir au courant des dernières menaces
- ☐ Se tenir au courant des derniers logiciels
- ☐ Pour maintenir les performances du système
- ☐ Pour améliorer la productivité du travail

- Q La surveillance des vulnérabilités de sécurité est essentielle pour rester à jour avec les dernières menaces et maintenir la sécurité d'une application par exemple.

p. 17 Solution n°3

Si nous reprenons notre cahier des charges, celui-ci demande de la sécurité dans son introduction mais le reste du document peut laisser présager que celle-ci sera oubliée, nous avons :

- Utilisation de formulaire : ces derniers devront être validés par le serveur pour éviter les attaques XSS.
- Connexion : celle-ci devra être faite en filtrant les entrées et avec l'utilisation d'un jeton CSRF afin d'éviter les attaques du même nom.
- Base de données SQL : les requêtes SQL sont des requêtes simples qui ne sont pas sécurisées.
- Rapidité : il sera nécessaire de tester la sécurité avant le déploiement afin de pouvoir corriger les éventuelles failles.

Des normes de développement avec la prise en compte des risques seront alors à effectuer.

p. 18 Solution n°4


Base de données SQL : il sera primordial de bien sécuriser les requêtes afin d'éviter les injections SQL notamment en utilisant des requêtes préparées par exemple.

Exercice p. 18 Solution n°5

Question 1

Qu'est-ce qu'une faille de sécurité ?


- ☐ Un défaut dans une théorie ou un argument
- ☒ Une erreur dans un système de sécurité
- ☐ Un logiciel obsolète
- ☐ Toutes ces réponses sont incorrectes

 Une faille de sécurité est une faiblesse, un défaut ou une erreur trouvée dans un système de sécurité qui a le potentiel d'être utilisée par un attaquant pour nuire à ce système. Elle peut se présenter sous forme de code logiciel défectueux ou de mauvaise configuration du système.

Question 2

Quel est le risque d'avoir des vulnérabilités dans une application ?


- ☒ Permettre des attaques automatisées
- ☒ Permettre à des logiciels malveillants et à des programmes non autorisés de fonctionner
- ☐ Il n'y a pas de risques
- ☐ Toutes ces réponses sont incorrectes

 Les vulnérabilités dans une application peuvent permettre diverses formes d'attaques, notamment les attaques automatisées et peuvent également donner la possibilité à des logiciels malveillants ou à des programmes non autorisés de fonctionner. Elles représentent un risque majeur pour la sécurité des applications.

Question 3

Comment se protéger efficacement contre les cyber-attaques ?

- ☐ En ignorant le problème
- ☒ En activant le 2FA
- ☐ En utilisant le même mot de passe pour tous vos comptes
- ☒ En créant les applications avec le concept de security by design

 Pour se protéger efficacement contre les cyber-attaques, il est recommandé d'activer l'authentification multifactorielle (le 2FA), et d'utiliser des mots de passe forts. Le concept de « *security by design* », qui intègre la sécurité dès la conception d'un système ou d'une application, est également une très bonne pratique pour prévenir les cyberattaques.

Question 4

Que signifie XSS ?


- ☐ Un type de cookie
- ☒ Cross-Site Scripting
- ☐ Le nouveau nom de twitter
- ☐ Xylophone Salulaire des Salamandres

 XSS signifie Cross-Site Scripting, il s'agit d'une faille basée sur l'exploitation de code comme le JavaScript.

Question 5

Qu'est-ce que la faille Cross-Site Request Forgery (CSRF) ?


- ☐ Une vulnérabilité qui permet d'injecter du code
- ☐ Une technique d'hameçonnage qui cible les e-mails des utilisateurs
- ☒ Une vulnérabilité qui force un utilisateur à exécuter des actions non désirées
- ☐ Toutes ces réponses sont incorrectes

 La faille Cross-Site Request Forgery (CSRF), ou attaque par contrefaçon de requête intersites, est une vulnérabilité qui permet à un attaquant de tromper un utilisateur pour qu'il exécute des actions non intentionnelles sur une application web où il est authentifié.

Question 6

Qu'est-ce qu'une faille d'accès dans une base de données ?


- ☐ Une faille qui permet de modifier les données sans autorisation
- ☒ Une faille qui permet à des attaquants non autorisés d'obtenir un accès à la base de données
- ☒ Une faille qui permet à un utilisateur d'accéder à des fonctionnalités auxquelles il ne devrait pas avoir accès
- ☐ Toutes ces réponses sont incorrectes

 Une faille d'accès dans une base de données se produit lorsque des utilisateurs non autorisés parviennent à accéder à des ressources sensibles de la base de données, ou lorsqu'un utilisateur accède à des fonctionnalités à lesquelles il n'est pas censé avoir accès.

Question 7

À quoi sert l'authentification à deux facteurs (2FA) ?


- ☒ À augmenter la sécurité des connexions aux comptes
- ☐ À simplifier le processus de connexion
- ☐ À récupérer les mots de passe oubliés plus facilement
- ☐ Toutes ces réponses sont incorrectes

 L'authentification à deux facteurs (2FA) sert à augmenter la sécurité des comptes en ajoutant une étape d'authentification supplémentaire. En plus du mot de passe, l'utilisateur doit fournir une autre preuve d'identité, ce qui rend beaucoup plus difficile pour un attaquant d'accéder au compte.

Question 8

Qu'est-ce que l'injection SQL ?

- ☒ Une technique d'attaque qui insère du code SQL
- ☐ Une technique de développement qui permet d'améliorer l'efficacité des requêtes SQL
- ☐ Une erreur de programmation courante lors de l'écriture des requêtes SQL
- ☐ Toutes ces réponses sont incorrectes

 L'injection SQL est une technique d'attaque qui consiste à insérer ou « injecter » des instructions SQL malveillantes dans une requête de base de données.