Audit de sécurité

Test d’intrusion SQL et XSS
Version auditée :

Rapport d’audit technique

26/01/2024

Auditeurs :

WANG PENGCHAO

Historique du document

Version	Auteur	Date	Commentaire
1	WANG PENGCHAO	26/01/2024	Document intermédiaire

Table des matières

1 - Démarche d'audit

Ce rapport d'audit a été élaboré dans le cadre d'une séance pratique du module R508 - Audits de sécurité, se présentant sous la forme d'une étude de cas réelle visant à simuler un rapport d'audit authentique. L'objectif de cet exercice est de favoriser une compréhension approfondie des processus et des pratiques liés à un audit de sécurité.

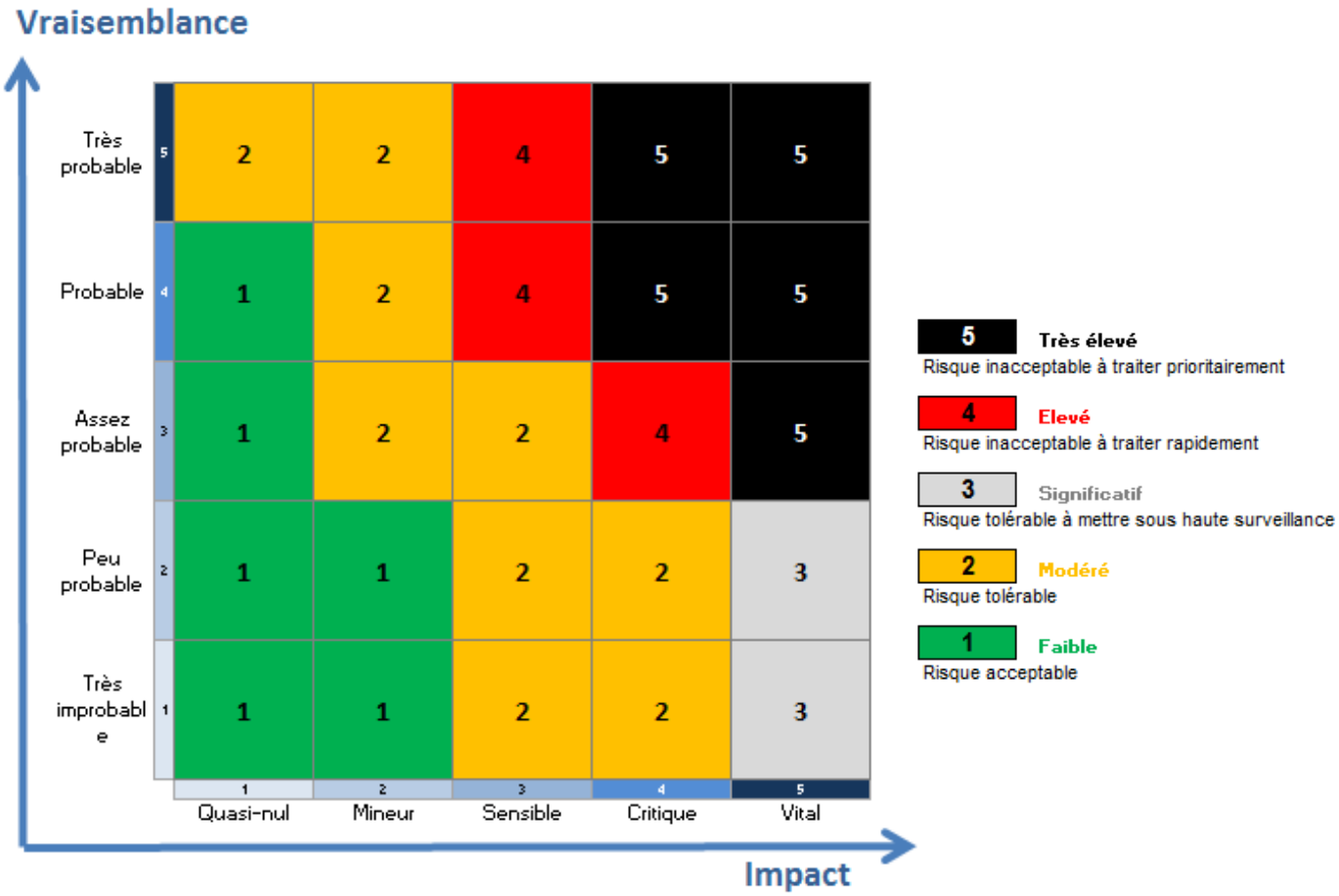
Il est essentiel de souligner l'importance cruciale des audits réguliers, car ils permettent de discerner à la fois les points forts et les vulnérabilités d'un système. Ils offrent ainsi une opportunité continue d'amélioration et de renforcement de la sécurité.

Ce TP se concentre particulièrement sur la compréhension et l'exploitation des vulnérabilités associées à une mauvaise configuration ainsi qu'à l'utilisation de langages SQL dans le contexte des applications web. L'objectif est d'approfondir les connaissances concernant les éventuelles failles de sécurité liées à ces aspects, ce qui renforce les compétences nécessaires pour identifier et résoudre efficacement de telles vulnérabilités au sein d'un environnement réel.

1.1 Organisation du document

Dans les prochaines sections de ce rapport, nous examinerons en détail les étapes distinctes de l'audit, à savoir la planification, la mise en œuvre et l'évaluation des résultats. Les conclusions résultant de cet audit joueront un rôle essentiel dans l'élaboration de recommandations pertinentes et dans la détermination des actions correctives requises.

1.2 Calcul de la criticité des vulnérabilités



2 - Listing des constats d'audit

2.1 Constat n°1 : Injection SQL

Description :

Dans un contexte particulier où une page est normalement conçue pour afficher les informations d'un utilisateur d'une "entreprise" lorsque l'identifiant utilisateur (user ID) est fourni, une manipulation simple a été exécutée avec succès. Cependant, cette manipulation a consisté à soumettre une requête alternative : " OR '1' = '1' -- ;' au lieu de fournir un ID utilisateur valide.

Cette requête a eu pour effet de contourner le système, ce qui a abouti à la récupération de toutes les informations sur les utilisateurs présents, sans nécessiter la fourniture d'un ID utilisateur valide. En exploitant cette vulnérabilité, nous avons pu afficher les informations de tous les utilisateurs, même en l'absence de connaissances préalables sur leur userID.

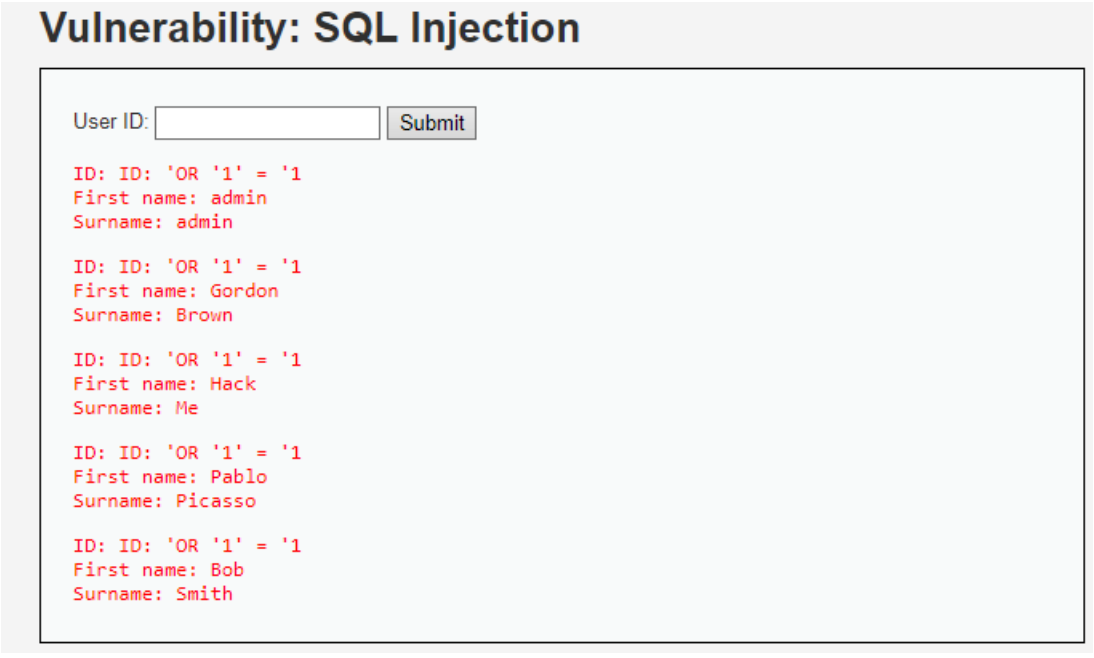
Cette démonstration met en lumière l'importance cruciale de sécuriser les entrées utilisateur pour éviter de telles manipulations et pour prévenir les accès non autorisés aux données sensibles.

Preuve :

Nous allons utiliser le langage SQL pour afficher les informations les infos sur tous les utilisateurs même si on ne connaît pas leur userID

L'opérateur logique OR est utilisé pour évaluer si au moins l'une des conditions est vraie. Dans le contexte de l'injection SQL, il peut être utilisé pour modifier le comportement d'une requête SQL en ajoutant une condition qui est toujours vraie, comme '1'='1'. Étant donné que '1' est toujours égal à '1', cette condition sera toujours vraie, ce qui signifie que la requête SQL modifiée retournera toujours des résultats, quels que soient les autres critères de la requête.

On peut apercevoir que des informations sont afficher comme le First name et Surname :



Recommendations :

- Validation et échappement des données : La validation rigoureuse des données utilisateur est fondamentale. Assurez-vous que toutes les données entrantes sont validées et échappées correctement avant d'être incluses dans une requête SQL. Utilisez des mécanismes de paramétrage de requête ou des requêtes préparées pour séparer les données de la requête SQL.

- Mises à jour régulières : Maintenez à jour votre système de gestion de bases de données, votre serveur web, votre application web et tous les frameworks utilisés. Les mises à jour contiennent souvent des correctifs de sécurité importants.

- Privilèges d'accès appropriés : Limitez les privilèges d'accès à la base de données. Les comptes utilisés par l'application web ne devraient avoir que les autorisations strictement nécessaires pour éviter que des attaquants n'aient un accès excessif en cas de compromission.

- Firewall d'application web (WAF) : Mettez en place un pare-feu d'application web (WAF) pour détecter et bloquer les attaques par injection SQL et d'autres attaques web courantes.

- Gestion des erreurs sécurisée : Évitez de révéler des informations sensibles lors d'erreurs. Personnalisez les messages d'erreur pour qu'ils ne divulguent pas d'informations sur la structure de la base de données.

- Contrôle des requêtes : Mettez en place des filtres pour les requêtes entrantes. Bloquez toute requête qui semble malveillante ou contient des caractères suspects.

- Audit des journaux : Surveillez les journaux d'activité de votre application web et de votre base de données pour détecter toute activité suspecte ou des tentatives d'injection SQL.

- Formation en sécurité : Formez les développeurs, les administrateurs et les équipes de sécurité à reconnaître et à prévenir les attaques par injection SQL. La sensibilisation à la sécurité est cruciale.

- Tests de sécurité réguliers : Effectuez des tests de sécurité réguliers, y compris des tests d'injection SQL, pour identifier et corriger les vulnérabilités avant qu'elles ne soient exploitées par des attaquants.

- Utilisation de pare-feu réseau : Configurez des pare-feu réseau pour bloquer le trafic non autorisé. Limitez l'accès aux serveurs de bases de données depuis l'extérieur de manière à ce que seules les machines nécessaires puissent s'y connecter.

Description :

Trouvez une requête pour afficher les informations sur la base de données

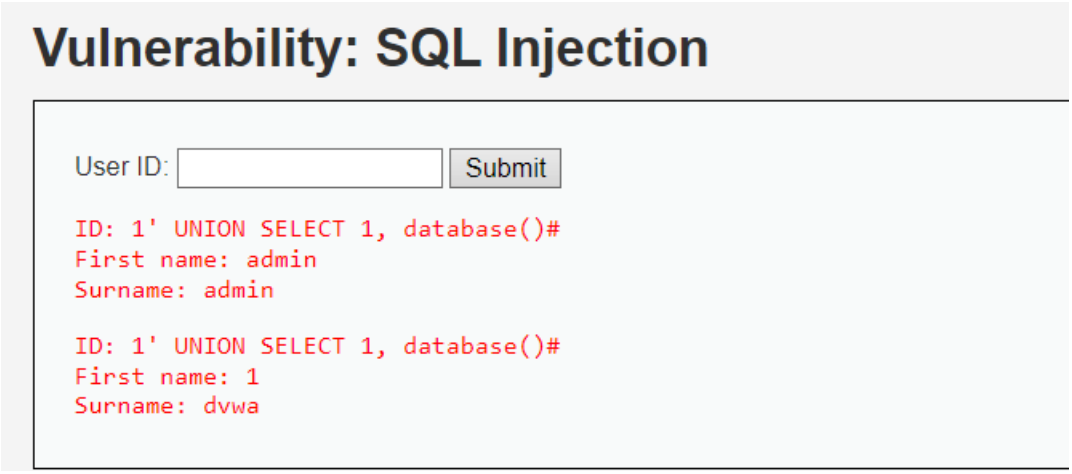
Preuve :

On utilisant la commande d'injection SQL ci-dessous, on peut afficher les informations sur la base de données :

Dans cette partie de la commande, "1'" est utilisé pour fermer la première partie de la requête SQL. Le caractère apostrophe (') est généralement utilisé pour délimiter les chaînes de caractères dans SQL.

UNION SELECT 1, database() : La partie "UNION SELECT 1, database()" est ajoutée à la fin de la première partie de la requête. L'opérateur "UNION" est utilisé pour combiner les résultats de deux requêtes distinctes en une seule réponse. Dans ce cas, deux colonnes sont sélectionnées : "1" et "database()". La première colonne "1" est simplement un nombre 1, tandis que la seconde colonne "database()" est utilisée pour extraire le nom de la base de données actuelle.

Le caractère "#" est utilisé pour commenter le reste de la requête. Cela permet d'ignorer tout ce qui suit dans la requête originale et d'éviter des erreurs.



Recommendations :

- Validation et échappement des données : La validation rigoureuse des données utilisateur est fondamentale. Assurez-vous que toutes les données entrantes sont validées et échappées correctement avant d'être incluses dans une requête SQL. Utilisez des mécanismes de paramétrage de requête ou des requêtes préparées pour séparer les données de la requête SQL.

- Mises à jour régulières : Maintenez à jour votre système de gestion de bases de données, votre serveur web, votre application web et tous les frameworks utilisés. Les mises à jour contiennent souvent des correctifs de sécurité importants.

- Privilèges d'accès appropriés : Limitez les privilèges d'accès à la base de données. Les comptes utilisés par l'application web ne devraient avoir que les autorisations strictement nécessaires pour éviter que des attaquants n'aient un accès excessif en cas de compromission.

- Firewall d'application web (WAF) : Mettez en place un pare-feu d'application web (WAF) pour détecter et bloquer les attaques par injection SQL et d'autres attaques web courantes.

- Gestion des erreurs sécurisée : Évitez de révéler des informations sensibles lors d'erreurs. Personnalisez les messages d'erreur pour qu'ils ne divulguent pas d'informations sur la structure de la base de données.

- Contrôle des requêtes : Mettez en place des filtres pour les requêtes entrantes. Bloquez toute requête qui semble malveillante ou contient des caractères suspects.

- Audit des journaux : Surveillez les journaux d'activité de votre application web et de votre base de données pour détecter toute activité suspecte ou des tentatives d'injection SQL.

- Formation en sécurité : Formez les développeurs, les administrateurs et les équipes de sécurité à reconnaître et à prévenir les attaques par injection SQL. La sensibilisation à la sécurité est cruciale.

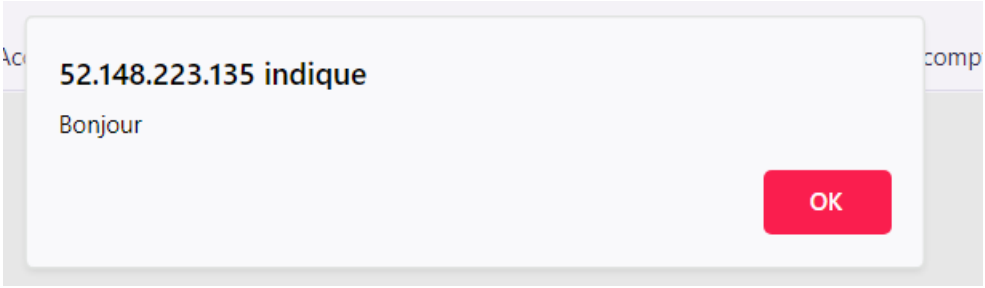
- Tests de sécurité réguliers : Effectuez des tests de sécurité réguliers, y compris des tests d'injection SQL, pour identifier et corriger les vulnérabilités avant qu'elles ne soient exploitées par des attaquants.

- Utilisation de pare-feu réseau : Configurez des pare-feu réseau pour bloquer le trafic non autorisé. Limitez l'accès aux serveurs de bases de données depuis l'extérieur de manière à ce que seules les machines nécessaires puissent s'y connecter.

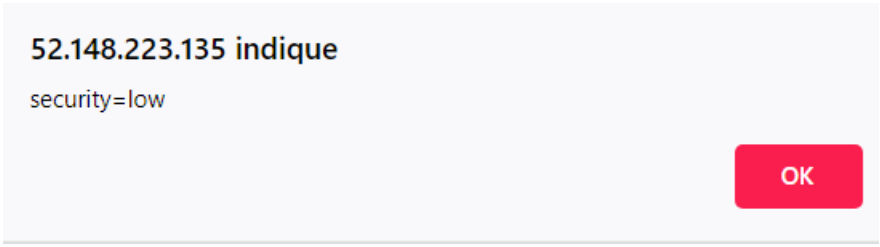
2.2 Constat n°2 :Attaque XSS

Description :
Déclenchez une fenêtre pop up avec une commande script

Preuve :
<script>alert('Bonjour');</script>'



Nous allons aussi afficher le cookie de session
<script>alert(document.cookie);</script>



Recommandation :

- Validation des données d'entrée : Validez et filtrez toutes les données d'entrée utilisateur pour vous assurer qu'elles ne contiennent pas de code malveillant. Utilisez des bibliothèques de validation et de nettoyage de données.

- Échappement des sorties : Échappez correctement toutes les données dynamiques affichées dans les pages web en utilisant des fonctions d'échappement appropriées pour le contexte (HTML, JavaScript, etc.). Cela empêche l'exécution de code malveillant.

- Utilisation de CSP (Content Security Policy) : Mettez en place une politique de sécurité des contenus (CSP) pour restreindre les sources autorisées à partir desquelles les scripts peuvent être chargés. Cela limite les possibilités d'exécution de scripts malveillants.

- Toujours utiliser HTTPS : Assurez-vous que votre site web utilise HTTPS pour chiffrer les communications entre le navigateur de l'utilisateur et le serveur. Cela aide à protéger les données sensibles et à prévenir les attaques de type Man-in-the-Middle (MITM).

- Mises à jour régulières : Gardez à jour votre application, les bibliothèques tierces, et les composants serveur pour bénéficier des correctifs de sécurité.

- Séparation du contenu et des données : Évitez de mélanger le contenu et les données. N'incorporez pas de données non échappées dans le code JavaScript, les URL, ou les balises HTML.

- Utilisation de bibliothèques sécurisées : Utilisez des bibliothèques et des frameworks web qui intègrent des protections contre les XSS, comme React, Angular, ou Vue.js.

- Éducation en sécurité : Formez votre équipe de développement à la sécurité web pour qu'elle comprenne les risques et les meilleures pratiques de sécurité.

- Tests de sécurité : Effectuez régulièrement des tests de sécurité automatisés et des audits de code pour détecter et corriger les vulnérabilités XSS.

- Rapport responsable : Encouragez les utilisateurs à signaler les vulnérabilités de sécurité qu'ils découvrent de manière responsable, afin que vous puissiez les corriger rapidement.

- Utilisation de bibliothèques de sécurité : Utilisez des bibliothèques de sécurité spécifiques pour votre langage de programmation, telles que OWASP ESAPI pour Java, pour renforcer la sécurité de votre application.