INJECTIONS : La vidéo montre une injection SQL en démontrant comment un attaquant peut manipuler un champ de message pour exécuter des commandes SQL non prévues. Cela pourrait entraîner des actions telles que la suppression de toutes les entrées de la base de données. Pour prévenir de telles attaques, l'utilisation de requêtes paramétrées est recommandée au lieu de concaténer directement des chaînes dans les instructions SQL.

Ensuite, la vidéo aborde l'injection JavaScript. En insérant une balise de script dans un message, un attaquant peut exécuter du JavaScript malveillant lorsque le message est affiché. Pour éviter cela, le script suggère de nettoyer les entrées avant de les stocker dans la base de données en utilisant des bibliothèques comme JSoup, qui peuvent supprimer le code potentiellement dangereux.

La vidéo insiste sur l'importance d'éviter la concaténation des entrées utilisateur dans les requêtes SQL et les sorties HTML. Elle recommande l'utilisation de frameworks et de bibliothèques qui gèrent automatiquement ces problèmes, comme les moteurs de templates (Thymeleaf) ou les frameworks front-end (Vue.js, Angular, React), qui possèdent des protections intégrées contre ces vulnérabilités. Adopter de bonnes pratiques de programmation aide à prévenir des attaques courantes mais facilement évitables.

DEFINITION :

Une injection SQL est une vulnérabilité de sécurité où un attaquant peut manipuler une requête SQL pour exécuter des commandes non prévues, souvent entraînant un accès ou une modification non autorisée des données. Une injection JavaScript implique l'insertion de code JavaScript malveillant dans une page web, qui s'exécute dans le navigateur de l'utilisateur, pouvant mener à un vol de données ou à d'autres actions malveillantes.

ACCESS CONTROL : Le problème initial est que les informations de l'utilisateur et les actions ne sont pas sécurisées, permettant à n'importe qui d'accéder et de modifier les données d'autres utilisateurs.

La première étape consiste à implémenter Spring Security pour authentifier les utilisateurs. Cependant, cette première implémentation ne vérifie pas suffisamment les permissions, permettant à des utilisateurs authentifiés d'accéder et de modifier les données d'autres utilisateurs.

La solution finale est d'améliorer le système en ajoutant des vérifications supplémentaires pour s'assurer que seuls les propriétaires des données peuvent les modifier. Cela est réalisé en utilisant des cookies d'authentification et en validant les actions des utilisateurs par rapport à leurs permissions dans le système.

DEFINITION ACCESS CONTROL :

Une vulnérabilité de contrôle d'accès se produit lorsqu'un système restreint incorrectement l'accès aux ressources, permettant ainsi à des utilisateurs non autorisés d'effectuer des actions ou d'accéder à des données qu'ils ne devraient pas pouvoir consulter. Cela peut entraîner des violations de données, des modifications non autorisées et d'autres problèmes de sécurité.

Voici les points clés concernant les vulnérabilités de contrôle d'accès :

Authentification incorrecte : Le système échoue à vérifier correctement l'identité des utilisateurs. Cela peut inclure des failles dans les mécanismes de connexion, tels que des mots de passe faibles, une gestion incorrecte des sessions ou l'absence de multi-authentification.

Autorisation insuffisante : Une fois authentifié, le système ne fait pas respecter correctement les permissions, permettant aux utilisateurs d'effectuer des actions au-delà de leurs privilèges prévus. Cela peut signifier accéder ou modifier des données, exécuter des commandes ou effectuer des fonctions administratives sans autorisation appropriée.

Escalade de privilèges horizontale et verticale :

Escalade horizontale : Un utilisateur régulier accède aux données ou aux fonctions d'autres utilisateurs au même niveau de privilège (par exemple, l'utilisateur A accède aux données de l'utilisateur B).

Escalade verticale : Un utilisateur régulier accède aux fonctions administratives ou à des fonctions de niveau supérieur (par exemple, l'utilisateur A obtient des privilèges administratifs).

Manipulation d'URL : Les attaquants peuvent manipuler les paramètres d'URL pour accéder à des zones ou des données restreintes. Si l'application ne valide pas correctement les permissions de l'utilisateur pour chaque requête, un attaquant pourrait accéder à des informations sensibles.

Références directes d'objet non sécurisées (IDOR) : Lorsque l'application référence directement des objets (comme des enregistrements de base de données) en utilisant des entrées fournies par l'utilisateur sans validation adéquate. Par exemple, si une URL contient un identifiant d'utilisateur ou un chemin de fichier, un attaquant pourrait modifier cette entrée pour accéder à des données qu'il ne devrait pas.

Listes de contrôle d'accès mal configurées (ACL) : Les ACL définissent quels utilisateurs ou processus système peuvent accéder aux ressources et quelles opérations ils peuvent effectuer. Des ACL mal configurées peuvent accidentellement accorder un accès plus large que prévu.

Exemple de scénario :

Considérez une application web qui permet aux utilisateurs de gérer leurs profils et de créer des quiz. Une vulnérabilité de contrôle d'accès pourrait survenir si l'application ne vérifie pas correctement la propriété d'un quiz avant de permettre sa modification. Un attaquant pourrait exploiter cela en changeant l'URL pour inclure un autre identifiant de quiz et modifier ou supprimer le quiz d'un autre utilisateur.

Conséquences réelles :

Violations de données : Des informations sensibles telles que des données personnelles, des dossiers financiers ou de la propriété intellectuelle pourraient être exposées.

Actions non autorisées : Les attaquants pourraient effectuer des actions qu'ils ne sont pas autorisés à faire, telles que changer les paramètres de configuration, supprimer des données ou exécuter des commandes.

Violations de conformité : Ne pas protéger les données peut entraîner des violations des lois et réglementations, ce qui peut entraîner des amendes et des actions judiciaires.

Dommages à la réputation : Les incidents de sécurité peuvent nuire à la réputation d'une organisation, entraînant une perte de confiance des clients et des affaires.

Mesures préventives :

Mettre en œuvre une authentification forte : Utiliser la multi-authentification et appliquer des politiques de mot de passe robustes.

Contrôle d'accès basé sur les rôles (RBAC) : Définir et appliquer soigneusement les rôles et les permissions, en veillant à ce que les utilisateurs aient l'accès minimum nécessaire.

Valider les entrées et les requêtes : S'assurer que les entrées et les requêtes des utilisateurs sont correctement validées et authentifiées à chaque étape.

Pratiques de développement sécurisé : Suivre des pratiques de codage sécurisé et auditer régulièrement le code pour détecter les problèmes de contrôle d'accès.

Tests de pénétration : Effectuer des tests de sécurité réguliers, y compris des tests de pénétration, pour identifier et corriger les vulnérabilités de contrôle d'accès.

Principe du moindre privilège : Toujours appliquer le principe du moindre privilège, en veillant à ce que les utilisateurs n'aient que l'accès nécessaire pour accomplir leurs tâches.

HTTPS : ### Résumé de la vidéo sur le protocole HTTPS

Dans cette vidéo, l'orateur explique le protocole HTTPS et son importance pour sécuriser les échanges sur le réseau. Voici les points clés abordés :

1. \*\*Introduction au HTTPS\*\* :

- HTTPS est une version sécurisée du protocole HTTP.

- Le "S" signifie "Secure" et indique que les communications sont cryptées pour protéger les données échangées.

2. \*\*Requête HTTP non sécurisée\*\* :

- Lorsqu'une requête HTTP est envoyée, plusieurs informations sont transmises en clair :

- Adresse IP de l'expéditeur.

- Type de navigateur (ex. : Chrome version 90).

- Système d'exploitation (ex. : Windows).

- Langue de l'utilisateur.

- Ces informations peuvent être visualisées dans les outils de développement de navigateurs comme Chrome.

3. \*\*Vulnérabilité des données non sécurisées\*\* :

- Les données envoyées via HTTP peuvent être interceptées par des tiers non autorisés.

- Un routeur compromis ou un attaquant sur le réseau peut facilement copier et lire ces données.

4. \*\*Protection offerte par HTTPS\*\* :

- HTTPS chiffre les données échangées, rendant les informations illisibles pour les tiers.

- Les seules informations non chiffrées sont l'adresse IP de l'expéditeur et le domaine du destinataire.

5. \*\*Exemple concret\*\* :

- Sans HTTPS, des informations sensibles (comme des mots de passe ou des données personnelles) peuvent être interceptées en clair.

- Avec HTTPS, même si un attaquant intercepte les données, il ne pourra pas les lire sans la clé de déchiffrement.

6. \*\*Utilisation d'un outil de sniffing de paquets\*\* :

- Un exemple avec un outil de sniffing (comme Wireshark) montre comment les données apparaissent en clair avec HTTP et sont chiffrées avec HTTPS.

### Conclusion

- L'utilisation de HTTPS est cruciale pour la sécurité des communications sur le web.

- HTTPS protège la confidentialité et l'intégrité des données échangées, réduisant les risques d'interception et de manipulation des informations par des attaquants.

### Note supplémentaire

Pour ceux qui souhaitent approfondir leurs connaissances, il est conseillé d'explorer les concepts de cryptographie et les mécanismes spécifiques de chiffrement utilisés par HTTPS, comme TLS (Transport Layer Security).

DEFINITION httpS :

HTTPS should be used because it encrypts data transmitted over the web, ensuring privacy, security, and protection against interception by malicious parties.

DEFINITION BLABLAH : La vidéo explique le fonctionnement de HTTPS, un protocole de sécurité sur internet. HTTPS crypte les données échangées pour assurer la confidentialité. Il utilise un chiffrement asymétrique avec des clés pour l'encodage et le décodage. La sécurité d'un site HTTPS est validée par un certificat qui confirme son identité. Toujours vérifier la présence du certificat pour s'assurer de la sécurité du site.

DEFINITION BLAH BLACH 2 HASHING :

La vidéo explique la différence entre l'encodage et le chiffrement, en se concentrant sur l'utilisation de ces méthodes dans le protocole HTTPS. L'encodage rend les données illisibles sans la clé appropriée, tandis que le chiffrement crypte les données de manière sécurisée. HTTPS utilise le chiffrement pour sécuriser les données échangées sur le réseau. Ensuite, le script aborde la gestion des mots de passe avec Spring Boot et Spring Security. Il recommande l'utilisation d'algorithmes de hachage pour sécuriser les mots de passe et explique comment configurer cela dans une application. Enfin, il encourage les auditeurs à explorer la configuration des encodeurs de mots de passe dans le serveur et à utiliser des outils pour accéder à la base de données.

Hashing is a process used in cryptography to convert input data of any size into a fixed-size string of characters, typically a hexadecimal number. This output string is called a hash value or hash code. Hash functions are designed to be fast and efficient, producing unique hash values for different inputs. Hashing is commonly used for data integrity verification, password storage, digital signatures, and various other security applications. It ensures that even a small change in the input data will produce a significantly different hash value, making it useful for detecting alterations or tampering. However, it's important to note that while hashing is a one-way process (you can't reverse the hash to obtain the original input), it's also not encryption, as the original data cannot be derived from the hash value.