BOU HANNA
Charbel
MCS 26.4

PROJET : CTF

---

1. **SQL Injection 1 :**

Search section of the find members page :

1 OR 1=1 UNION SELECT NULL—

FLAG : 10316d8347b1e4968b25c4c40fe24ed9e44dcea7080987c8392ba031ff5

Comment vous protéger :

Utilisez des requêtes préparées (Prepared Statements). Les requêtes préparées garantissent qu'un attaquant ne peut pas modifier l'intention d'une requête, même si des commandes SQL sont insérées par l'attaquant. Utilisez des procédures stockées (Stored Procedures). Elles obligent le développeur à simplement construire des instructions SQL avec des paramètres qui sont automatiquement paramétrés, à moins que le développeur ne fasse quelque chose de largement inhabituel. La différence entre les requêtes préparées et les procédures stockées est que le code SQL d'une procédure stockée est défini et stocké dans la base de données elle-même, puis appelé depuis l'application. Ces deux techniques ont la même efficacité pour prévenir l'injection SQL, votre organisation devrait donc choisir l'approche qui a le plus de sens pour vous.

2. **SQL Injection 2 :**

FLAG : f2a29020ef3132e01dd61df97fd33ec8d7fcd1388cc9601e7db691d17d4d6188

Comment vous protéger :
Pareil que le 1.

3. **Cross-Site Request Recovery :**

HTTP header manipulation :

- **Referer :** https://www.nsa.gov/
- **User-Agent :** ft_bornToSec

FLAG :
F2A29020EF3132E01DD61DF97FD33EC8D7FCD1388CC9601E7DB691D17D4D6188

Comment vous protéger :

Ne laissez pas de commentaires sensibles dans le code source que vous ne voulez pas voir rendus publics ! Vérifiez également côté serveur si l'en-tête origin/referer est présent et si sa valeur correspond à l'origine cible. Créez une vérification stricte pour le referer et créez une liste blanche (whitelist) pour l'agent utilisateur.

4. **Unvalidated Redirect and Forward attack**

URL Forge : ''index.php?page=redirect&site=malicious_link''

FLAG :
B9E775A0291FED784A2D9680FCFAD7EDD6B8CDF87648DA647AAF4BBA288BCAB3

Comment vous protéger :

La meilleure solution est de ne pas utiliser de redirections ou de transferts (forwards), si ce n'est pas un aspect commercial crucial du site web. Vous pourriez également stocker les URL complètes dans la base de données, leur attribuer des identifiants et utiliser ces identifiants comme paramètres de requête. Avec une telle approche, les attaquants ne pourront pas rediriger ou transférer vers des pages non autorisées. Vous pouvez également créer une liste blanche (whitelist) d'URL que vous considérez comme sûres pour une redirection. Cette solution est cependant risquée, car des erreurs dans le filtrage peuvent rendre certains vecteurs d'attaque possibles.

5. **Web Parameter Tampering**

Request Payload : sujet=42&valeur=42

FLAG :
03A944B434D5BAFF05F46C4BEDE5792551A2595574BCAFC9A6E25F67C382CCAA

Comment vous protéger :

L'utilisation d'expressions régulières (regex) pour limiter ou valider les données peut aider à réduire cette vulnérabilité, ou en évitant d'inclure des paramètres dans la chaîne de requête. Utilisez également une validation côté serveur pour comparer les données avec toutes les entrées.

6. **Path Traversal Attack**

Search Member : 192.168.1.210/?page=../../../../etc/shadow

/etc/passwd : 192.168.1.210/?page=../../../../../../../etc/passwd

FLAG :
b12c4b2cb8094750ae121a676269aa9e2872d07c06e429d25a63196ec1c8c1d0

Comment vous protéger :

Privilégiez le travail sans saisie utilisateur lors de l'utilisation d'appels au système de fichiers

- Utilisez des index plutôt que des portions réelles de noms de fichiers lors de l'utilisation de templates ou de fichiers de langue (par exemple, la valeur 5 provenant de la soumission utilisateur = Tchécoslovaque, plutôt que d'attendre que l'utilisateur renvoie "Tchécoslovaque")
- Assurez-vous que l'utilisateur ne peut pas fournir toutes les parties du chemin – entourez-le avec votre code de chemin
- Validez les entrées de l'utilisateur en n'acceptant que ce qui est connu comme valide – ne vous contentez pas de nettoyer les données
- Utilisez des environnements cloisonnés (chrooted jails) et des politiques de contrôle d'accès au code pour restreindre l'endroit où les fichiers peuvent être obtenus ou sauvegardés
- Si vous êtes forcé d'utiliser des entrées utilisateur pour des opérations sur les fichiers, normalisez les données avant de les utiliser dans les API d'entrée/sortie de fichiers, comme normalize()

Il s'agit de bonnes pratiques de sécurité pour prévenir les vulnérabilités de type "Path Traversal" ou "Directory Traversal" qui permettent aux attaquants d'accéder à des fichiers non autorisés sur le système.