Rapport — TP projet (CTF)

Stored XXS :

FLAG :
0FBB54BBF7D099713CA4BE297E1BC7DA0173D8B3C21C1811B916A3A86652724E

En testant une page de commentaires, j'ai constaté qu'elle était vulnérable à une attaque stored XXS. Initialement, il semble que les tags HTML soient filtrés, mais après quelques tentatives, j'ai pu contourner ce filtre en injectant un tag IMG malformé contenant du code JavaScript, ce qui a permis d'exécuter une alerte XXS.

En actualisant la page, j'ai remarqué que le script injecté était constamment réexécuté, ce qui montre que le script malveillant est stocké sur le serveur (dans une base de données, par exemple) et se déclenche à chaque demande de la victime.

Comment s'en protéger : Il est nécessaire d'effectuer une validation d'entrée stricte avec des expressions régulières ou des bibliothèques de confiance, ainsi que de "nettoyer" (sanitiser) les données avant de les stocker.

Unvalidated Redirect and Forward Attack :

FLAG :
B9E775A0291FED784A2D9680FCFAD7EDD6B8CDF87648DA647AAF4BBA288BCAB3

Lors de l'exploration du site, j'ai remarqué qu'il n'y avait pas de validation pour les redirections vers les réseaux sociaux en bas de la page. Les redirections ouvertes ne sont pas directement critiques pour le site lui-même et ne permettent pas à un attaquant de voler des données du propriétaire du site, mais elles sont dangereuses pour les utilisateurs. Elles sont souvent utilisées dans des attaques de phishing : l'attaqué reçoit un email légitime avec un lien pointant vers un domaine attendu. Cependant, en réalité, le lien peut contenir des paramètres qui redirigent vers un autre site malveillant.

En testant cela sur le site, j'ai forgé l'URL et cliqué sur le lien des réseaux sociaux, ce qui m'a permis de récupérer le flag numéro 3.

Comment corriger la vulnérabilité : La meilleure solution est d'éviter d'utiliser des redirections ou des forwards, sauf si cela est crucial pour le site. Sinon, vous pouvez stocker les URL complètes dans la base de données, leur attribuer des identifiants et utiliser ces identifiants comme paramètres dans les requêtes. Ainsi, les attaquants ne pourront pas manipuler les redirections. Une autre option consiste à autoriser certaines URL dans une liste blanche, mais cette approche présente des risques si le filtrage est mal implémenté.

Password Guessing Attack :

FLAG :
B3A6E43DDFF8B4BBB4125E5E723040433827759D4DE1C04EA63907479A80A6B2

La page de connexion utilise une requête GET pour envoyer les identifiants, ce qui expose les données en texte clair, capturables par un attaquant via un sniffing de réseau. De plus, il n'y a pas de limite sur le nombre de tentatives depuis la même IP, ce qui permet une attaque par force brute. J'ai intercepté la requête et utilisé Burp Suite avec une liste de noms d'utilisateur et de mots de passe pour obtenir une combinaison valide et récupérer le flag numéro 10.

Solution : Pour éviter les attaques par force brute, verrouillez les comptes après plusieurs tentatives infructueuses et ajoutez une vérification avec une question secrète après quelques erreurs. Évitez d'utiliser GET pour des informations sensibles et validez les entrées des formulaires.

Web Parameter Tampering :

FLAG :
03A944B434D5BAFF05F46C4BEDE5792551A2595574BCAFC9A6E25F67C382CCAA

En explorant le site, j'ai découvert la page de sondage et observé dans l'onglet réseau de la console qu'une requête était envoyée au serveur. C'est normal, mais que se passe-t-il si je manipule les paramètres échangés entre le client et le serveur ?

J'ai intercepté la requête POST avec Burp Suite, modifié les paramètres, puis renvoyé la requête. La manipulation a réussi, et j'ai récupéré le flag numéro 4 !

Comment corriger la vulnérabilité : Utiliser des expressions régulières (regex) pour limiter ou valider les données et éviter d'inclure des paramètres dans la chaîne de requête. Il est également crucial de faire une validation côté serveur pour comparer les données avec toutes les entrées.