RATTRAPAGE ANTOINE BAYSSAC

Sécuriser un site web

Dans ce travail, nous allons analyser trois types d'attaques possibles sur le site ou l'application de l'entreprise Picard, en expliquant les méthodes utilisées pour les exécuter et les meilleures pratiques à adopter pour les contrer efficacement.

La première méthode sont les injections SQL

Une attaque par injection SQL consiste à insérer une requête SQL malveillante dans les champs de saisie d'un formulaire ou directement dans l'URL d'une application.

Par exemple, un attaquant peut entrer un code SQL tel que `'; DROP TABLE utilisateurs; --` dans un champ de saisie.

On pourrait penser que dans ce formulaire de chez picard on pourrait faire une Injection SQL

Si l'application ou le site de picard pour l'exemple ne filtre pas ou n'échappe pas correctement les entrées utilisateur, cette requête sera exécutée par le serveur, ce qui va provoquer une suppression ou une modification non autorisée des données.

Il existes différentes méthodes afin d'empêcher cette situation

- Utiliser des **requêtes préparées** qui séparent les données des commandes SQL, empêchant ainsi l'injection de code malveillant.
- Limiter les **droits des utilisateurs SQL** de sorte qu'un compte compromis ne puisse pas accéder ou modifier des données critiques.
- Valider et échapper les entrées utilisateur côté serveur.

La deuxième méthode sont les **Attaque XSS (Cross-Site Scripting),** elle consiste à injecter du code malveillant, souvent du JavaScript, dans une page web qui sera ensuite affichée aux utilisateurs. Ce code peut être inséré via des formulaires, des champs de commentaires, ou d'autres points d'entrée sur le site.

Lorsque d'autres utilisateurs accèdent à la page contenant le script injecté, il va s'exécuter dans leur navigateur avec les mêmes droits que le site visité.

Un exemple pertinent de point d'entrée pour une attaque XSS sur le site de Picard pourrait être le **formulaire de recherche** ou **les champs de commentaires/revues de produits**. Exemple le site permet aux utilisateurs de rechercher des produits comme ci dessous.

Un utilisateur malveillant pourrait essayer d'injecter un script malveillant dans le champ de recherche. Par exemple, au lieu de taper un terme de recherche standard, l'utilisateur pourrait entrer une balise script comme

```
<script>alert('XSS')</script>
```

Si picard ne filtre pas correctement cette entrée, ce script serait injecté dans la page de résultats de recherche et exécuté dans le navigateur des autres utilisateurs qui consultent cette page, déclenchant l'alerte définie par l'attaquant.

Les risques sont forts

- Vol de cookies, permettant à l'attaquant de se faire passer pour l'utilisateur légitime.
- Accès aux données sensibles stockées par le navigateur.
- Redirection des utilisateurs vers des sites malveillants.
- Réécriture ou modification du contenu affiché sur la page web.
- Exécution d'actions à l'insu de l'utilisateur, comme l'envoi de formulaires ou l'exécution de transactions.

Plusieurs solutions s'offrent à nous pour contrer ce genre d'attaques :

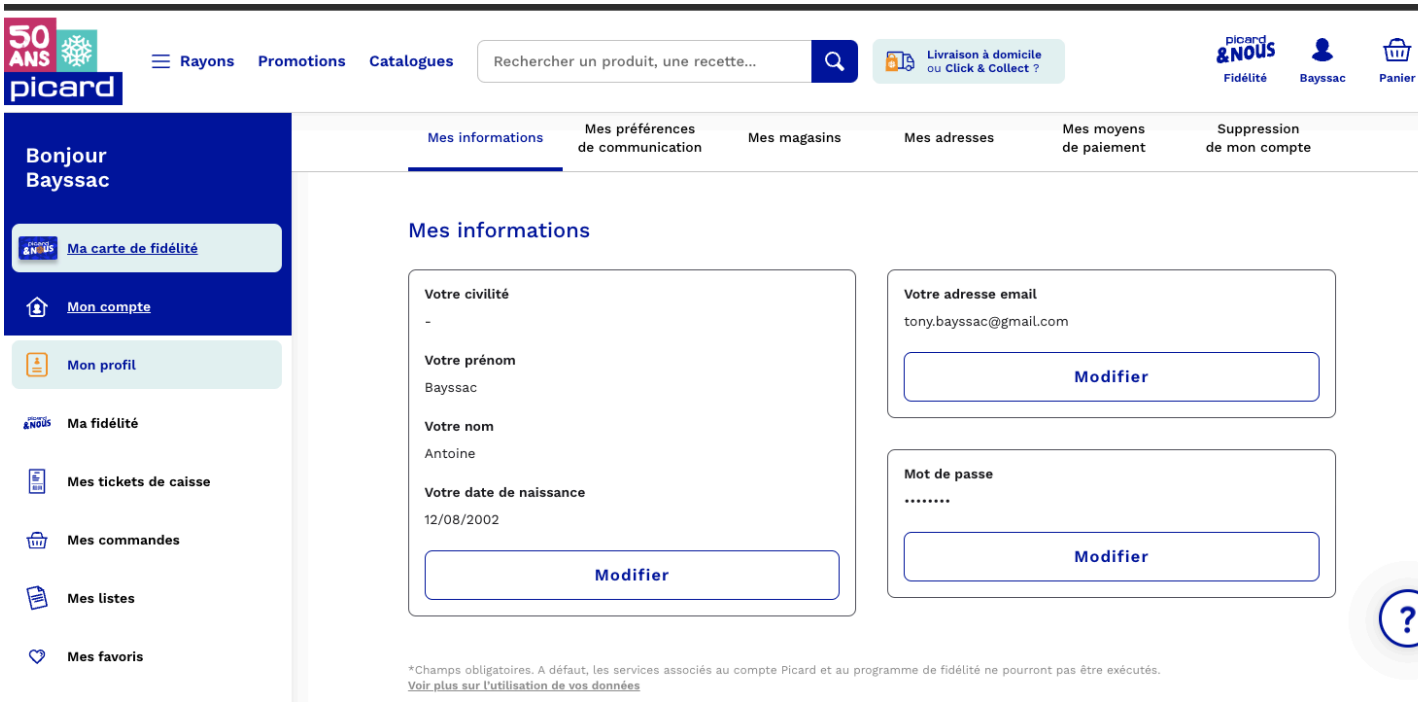
> **Échapper les caractères spéciaux :**
> On pourrait traiter toutes les entrées utilisateur en échappant les caractères spéciaux qui pourraient être interprétés comme du code HTML ou JavaScript. Par exemple, en PHP, on peut utiliser la fonction `htmlspecialchars()` pour convertir les caractères spéciaux (`<`, `>`, `&`, `"`, `'`) en entités HTML, empêchant ainsi leur interprétation par le navigateur.
>
> ou encore **la Validation côté serveur** ou Il est impératif de valider toutes les données entrantes côté serveur, car les données qui proviennent du client ne sont pas dignes de confiance.
>
> Cela inclut aussi la vérification de la longueur, du type et du contenu des champs pour s'assurer qu'ils ne contiennent pas de code malveillant.

En résumé, grâce à ces pratiques, on peut prévenir l'exécution de scripts non autorisés.

Et enfin la troisième attaque possible sur le site de picard serait **l'Attaque CSRF (Cross-Site Request Forgery) qui** vise à tromper un utilisateur authentifié pour qu'il effectue des actions non désirées sur une application web. Cette attaque exploite la confiance qu'une application web accorde à l'utilisateur.

Le site de Picard dispose d'une fonctionnalité permettant aux utilisateurs connectés de gérer leur compte, y compris de modifier leur adresse de livraison ou de changer leur mot de passe.

Vous pouvez voir que je me suis créé un compte chez picard et on peut voir qu'ils ont une fonctionnalité qui permet de modifier ses informations de compte.



Je pourrais donc en tant qu'attaquant créer un site web malveillant ou envoyer un email avec une image ou un lien spécialement conçu pour exploiter une faille CSRF sur le site de Picard.

Supposons que l'URL pour changer l'adresse de livraison sur le site de Picard ressemble à ceci :

```
https://www.picard.fr/compte/modifier_adresse?adresse=attacker_address
```

Je pourrais inclure ce lien dans une balise `<img>` dans un email ou sur une page web :

```
<img src="https://www.picard.fr/compte/modifier_adresse?adresse=123 Fausse Adresse">
```

Si un utilisateur de Picard, déjà authentifié sur le site, clique sur le lien que j'ai envoyé par mail ou visite la page contenant cette image, la requête sera automatiquement envoyée au serveur Picard avec les cookies d'authentification de l'utilisateur. Le serveur pourrait alors traiter cette requête comme légitime, changeant l'adresse de livraison de l'utilisateur à l'insu du client.

Cette attaque pourrait me permettre de rediriger les livraisons vers une adresse sous son contrôle, ce qui pourrait être utilisé pour voler des commandes de la vitcime.

Enfin pour contrer cette attaque sur le site de Picard, il devrait utiliser des **jetons CSRF :**

Lorsqu'un utilisateur souhaite changer son adresse de livraison, le formulaire de modification doit inclure un **jeton CSRF**. Ce jeton est une valeur unique et aléatoire, générée par le serveur à chaque fois que le formulaire est chargé.

C'est-à- dire que si l'utilisateur soumet le formulaire, le serveur vérifie que le jeton CSRF fourni correspond à celui généré pour la session en cours. Si le jeton est absent ou incorrect ou a expiré, la requête est rejetée, ce qui va empêcher l'attaquant de forcer une modification de l'adresse via une requête malveillante.

Un autre moyen de contrer serait d'utiliser un **Attribut SameSite dans les cookies** ou on configure les cookies d'authentification utilisés par le site avec l'attribut **SameSite=Strict**. Cela garantit que les cookies ne seront pas envoyés avec des requêtes initiées par des sites tiers, comme ceux qui pourraient être intégrés dans un email pour suivre notre exemple.

Du coup avec l'attribut SameSite, même si un utilisateur authentifié visite un site malveillant ou clique sur un lien suspect, les cookies d'authentification nécessaires pour exécuter une action sur le site de Picard ne seront pas envoyés, et la tentative d'attaque échouera.

On peut aussi rajouter une **Validation supplémentaire côté serveur** qui va faire en sorte que des actions critiques comme le changement d'adresse de livraison, on va ajouter une validation supplémentaire côté serveur en demandant à l'utilisateur de saisir à nouveau son mot de passe avant d'autoriser la modification. Cela garantit qu'une action aussi risquée ne peut être réalisée qu'avec une authentification réelle et volontaire de l'utilisateur.

Conclusion :

A travers cette analyse, nous avons exploré trois types d'attaques courantes sur les sites web et applications, à savoir les injections SQL, les attaques XSS (Cross-Site Scripting), et les attaques CSRF (Cross-Site Request Forgery). On s'est basé sur le site de l'entreprise Picard, on a identifié des exemples hypothétiques de points d'entrée pour ces attaques, tels que les champs de recherche ou les formulaires de gestion de compte.

Je tiens quand même à préciser que ces exemples ne sont pas forcément réalisables sur le site actuel de Picard, mais ils montrent des scénarios typiques que des attaquants pourraient exploiter si les bonnes pratiques de sécurité n'étaient pas respectées.

Cet exercice montre l'importance cruciale de la sécurité dans le développement web, où chaque fonctionnalité exposée à l'utilisateur peut devenir une potentielle porte d'entrée pour une attaque. En intégrant des pratiques telles que l'échappement des entrées utilisateur, la validation côté serveur, l'utilisation de jetons CSRF, et d'autres mesures, les entreprises comme Picard peuvent fortement réduire les risques de voir leur site compromis, ce qui réduirait ainsi le risques d'attaques pour leur site et protégerait les informations de leurs utilisateurs et leurs données sensibles.

Source :

Attaques Injections SQL : https://owasp.org/www-community/attacks/SQL_Injection

Attaques XSS : https://owasp.org/www-community/attacks/xss/

Attaques CSRF : https://owasp.org/www-community/attacks/csrf