Sécurité Web

Yannick Mazières

Erreur de débutant – Exemple + solution

Rodrigue

Rodrigue est authentifié au système et veut effacer un commentaire lui appartenant. Une requête **Post** est envoyée au serveur pour supprimer le commentaire **blog.com/comment/delete?id=10**

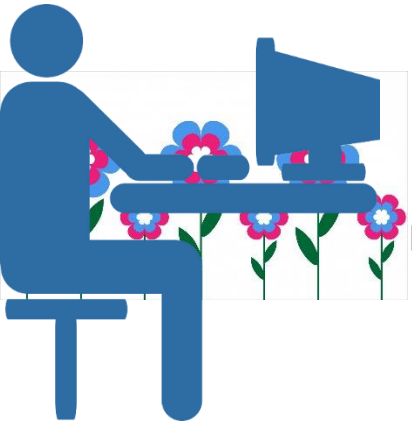
Pour corriger le problème, avant de supprimer un commentaire il faut vérifier que l'utilisateur est authentifié est qu'il est le propriétaire.

Yolande

Yolande est authentifiée au système et utilise un outil comme **fiddler** pour envoyer la requête **Post blog.com/comment/delete?id=10**

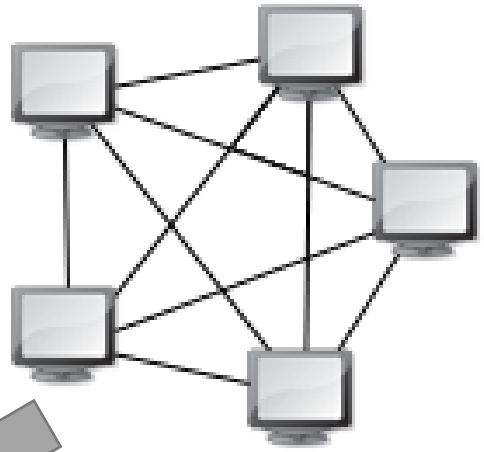
Mot de passe en clair – Problème

Rodrigue



1

Rodrigue veut s'authentifier. Une requête **Post** avec mot de passe en clair est envoyée au serveur.



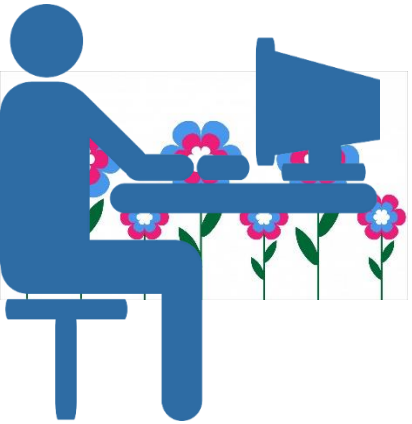
2

Yolande écoute le trafic sur le réseau avec un outil comme Wireshark

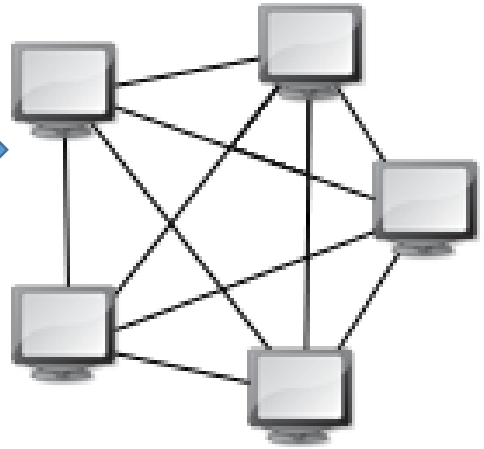
Yolande

Mot de passe en clair – Solution (utiliser HTTPS)

Rodrigue



Toutes les communications entre le navigateur et le site web sont cryptées



Yolande écoute le trafic, mais ne peut voir que des données cryptées.

Yolande

Avec **MVC.Net**, il faut ajouter dans Global.asax la ligne **GlobalFilters.Filters.Add(new RequireHttpsAttribute());**
Bien entendu, il faut aussi configurer le serveur et avoir un certificat SSL.

Données sensibles non cryptées – problème + Solution

Yolande a réussi, car elle est très forte, à avoir accès à la BD. Elle a accès à toutes les données sensibles (mots de passe, cartes de crédit, etc.)

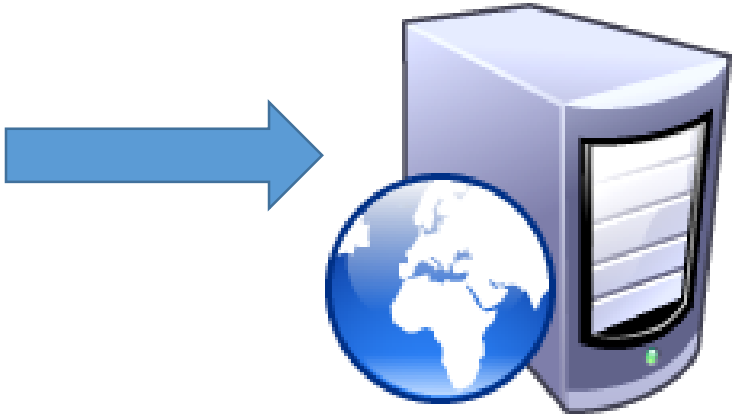
Pour corriger le problème, crypter dans la BD toutes les données sensibles.

Plusieurs librairies existent pour faire le travail de cryptage (exemple BCrypt.net)

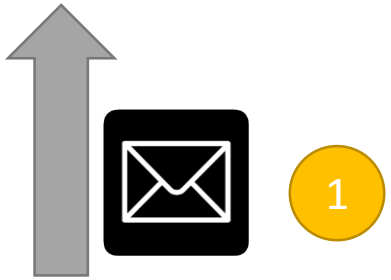
Cross-Site Request Forgery (CSRF) – Le problème

Rodrigue

Rodrigue qui est authentifié au système lit le courriel de Yolande. Le navigateur tente de récupérer l'image (qui n'existe pas) mais c'est en réalité le lien web qui est exécuté par le navigateur.



Luc est authentifié et est le propriétaire du message. La demande de suppression est donc exécutée !



Yolande envoie un courriel à Rodrigue avec un lien sur une image qui n'existe pas.

…

<img src="http:// blog.com/message/delete?id=10 " width="0" height="0" border="0">

…

Yolande

Pour simplifier l'exemple, c'est une requête **Get** qui est utilisée au lieu d'un **Post**.

On pourrait penser corriger le problème en utilisant un **Post**, mais il existe plusieurs astuces pour y arriver aussi avec un **Post**.

Cross-Site Request Forgery (CSRF) – La Solution avec MVC.Net

Rodrigue

3

Rodrigue est authentifié et veut effacer un commentaire lui appartenant.

La vue contient maintenant un **Jeton** unique (qui a été créé sur le serveur avec **@Html.AntiForgeryToken()**) et est ensuite renvoyée par la requête.

1 Dans le **contrôleur**, décorer la méthode avec l'attribut **[ValidateAntiForgeryToken]**

2 Dans la **vue**, faire un appel à **@Html.AntiForgeryToken()**

4 La méthode décorée avec **[ValidateAntiForgeryToken]** s'assure que le jeton (envoyé lors de la requête) est le bon.

Cross-Site Request Forgery (CSRF) – La solution avec MVC.Net

Rodrigue

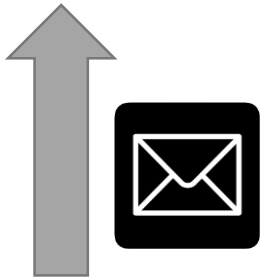
2

Rodrigue qui est authentifié au système lit le courriel de Yolande. Le navigateur tente de récupérer l'image (qui n'existe pas) mais c'est en réalité le lien web qui est exécuté par le navigateur.

La vue ne contient pas de Jeton unique, donc la requête est envoyée sans Jeton.

3

La méthode décorée avec **[ValidateAntiForgeryToken]** s'assure que le jeton (envoyé lors par la requête) est le bon. Aucun Jeton n'est contenu dans la requête. L'accès est impossible.



1

Yolande envoie un courriel à Rodrigue avec un lien sur une image qui n'existe pas.

…

<img src="http:// blog.com/message/delete?id=10 " width="0" height="0" border="0">

…

Yolande