Exercice de Cross Site Scripting

# Création du projet

Pour cet exercice, nous repartirons de l’exercice créé pour tester les injections SQL, avec la gestion des personnes en application Blazor Web Assembly.

Attention, pour que l’exercice fonctionne, il faut que le problème d’injection ait été résolu à l’aide des SqlParameters. La commande d’insertion doit se trouver comme suit :

var commande = conn.CreateCommand();

commande.CommandText = "INSERT INTO PERSONNES (nom, prenom, age) VALUES (@nom, @prenom, @age)";

commande.Parameters.Add(new SqliteParameter("nom", personne.Nom));

commande.Parameters.Add(new SqliteParameter("prenom", personne.Prenom));

commande.Parameters.Add(new SqliteParameter("age", personne.Age));

commande.ExecuteNonQuery();

Et la commande de lecture permet de montrer une autre grammaire plus courte :

var commande = conn.CreateCommand();

commande.CommandText = "SELECT nom, prenom, age FROM PERSONNES WHERE nom LIKE @pattern";

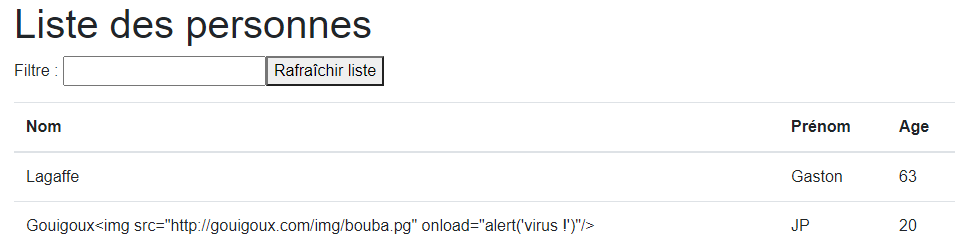
commande.Parameters.AddWithValue("pattern", "%" + IndicationNom + "%");

# Première tentative

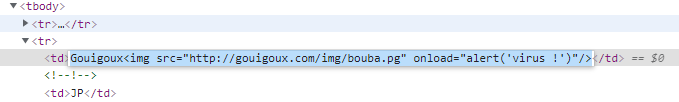
Dans un premier temps, nous allons tenter une approche naïve de XSS en essayant de faire en sorte que, lorsque le nom de famille s’ajoute dans le tableau, le logiciel exécute un script parasite. Pour cela, nous ajoutons à un nom quelconque la chaîne suivante :

<img src="http://gouigoux.com/img/bouba.pg" onload="alert('virus !')"/>

Le système de binding est bien fait dans Blazor, car il nous protège par défaut de ce type d’attaque, et si on essaie d’exploiter une faille XSS, le mécanisme d’affichage montre que le contenu est pris comme texte et non exécuté :



Un coup d’œil dans l’explorateur d’éléments de la page montre que le contenu textuel complet correspond à la tentative d’injection d’HTML :



Et en sélectionnant le HTML, on voit que le contenu a été encodé à la volée, ce qui rend l’attaque inopérante :

<td>Gouigoux&lt;img src="http://gouigoux.com/img/bouba.pg" onload="alert('virus !')"/&gt;</td>

C’est un premier enseignement de cet exercice que le fait d’utiliser des stacks logicielles bien codées et pas antéchristiques permet de réduire les problèmes à la racine, et ce sans le moindre effort, ce qui constitue un excellent premier rideau de sécurité.

Malheureusement, le code à produire est parfois plus complexe ou nécessite de mettre en place du code qui rend les failles XSS présentes.

# Faille XSS

Imaginons par exemple que le code serveur permette de générer une page de résumé de la personne sélectionnée par son nom, avec le code ci-dessous à rajouter dans le contrôleur :

[HttpGet("fiche")]

public ContentResult GenererFiche([FromQuery] string nom)

{

StringBuilder sb = new StringBuilder();

sb.AppendLine("<html>");

sb.AppendLine("<body>");

using (var conn = new SqliteConnection("Data Source=test.db"))

{

conn.Open();

var commande = conn.CreateCommand();

commande.CommandText = "SELECT prenom, age FROM PERSONNES WHERE nom=@nom";

commande.Parameters.Add(new SqliteParameter("nom", nom));

using (var reader = commande.ExecuteReader())

{

if (reader.Read())

{

sb.Append("<h1>").Append(reader.GetString(0)).Append(" ").Append(nom).AppendLine("</h1>");

sb.Append("<p>Agé.e de ").Append(reader.GetInt32(1).ToString()).AppendLine(" ans</p>");

}

else

{

sb.Append("<h1>").Append(nom).Append(" ne fait pas partie de notre annuaire !").AppendLine("</h1>");

}

}

}

sb.AppendLine("</body>");

sb.AppendLine("</html>");

return Content(sb.ToString(), "text/html", Encoding.UTF8);

}

Ne pas oublier de rajouter le using System.Text; pour que le résultat compile.

Le fait d’insérer directement le nom dans le code HTML généré dans le else crée une faille de type Cross Site Scripting, car on va pouvoir injecter à la volée du code parasite. Par exemple une image qui, en se chargeant, va rediriger vers un autre site pirate (code Javascript sur onload='window.location = …'), sur lequel on aura par exemple repris la page de Google ou d’une banque pour récupérer les mots de passe.

# Faille XSS rémanente

La faille rémanente est plus problématique car le fait d’envoyer ce genre de code parasite sur la base de données fait que toutes les autres personnes qui afficheront la fiche seront attaquées.

A titre d’indice, le prénom est sensible à cette attaque rémanente. Le but de l’exercice est de faire afficher une fenêtre modale en Javascript avec un message d’alerte (fonction alert).