Cours .Net
Sécurité

Lemettre Arnaud
Arnaud.lemettre@gmail.com

EPITA
ÉCOLE D'INGÉNIEURS EN INFORMATIQUE

Microsoft® .NET

Sommaire

- Utilisation du profile
- Authentification
- Autorisation
- Construction automatisée VS manuelle

Introduction

- La sécurité dans une application web est l'une des plus grosses dépenses en terme de temps et de budget. C'est pour cela que Microsoft nous fournit des blocs de sécurité réutilisables.

PROFILE

Profile

- Le profil utilisateur en asp.net est un ensemble de propriété associé à une personne qui visite votre site.
- Dans ces informations on peut retrouver :
  - Les préférences au niveau des couleurs
  - L'adresse
  - Et tout autre information que vous souhaitez tracer.

Le profile n'est disponible que pour un projet de type SiteWeb et non pas WebApplication. Ceci vient de la gestion différente par le framework de ces projets.

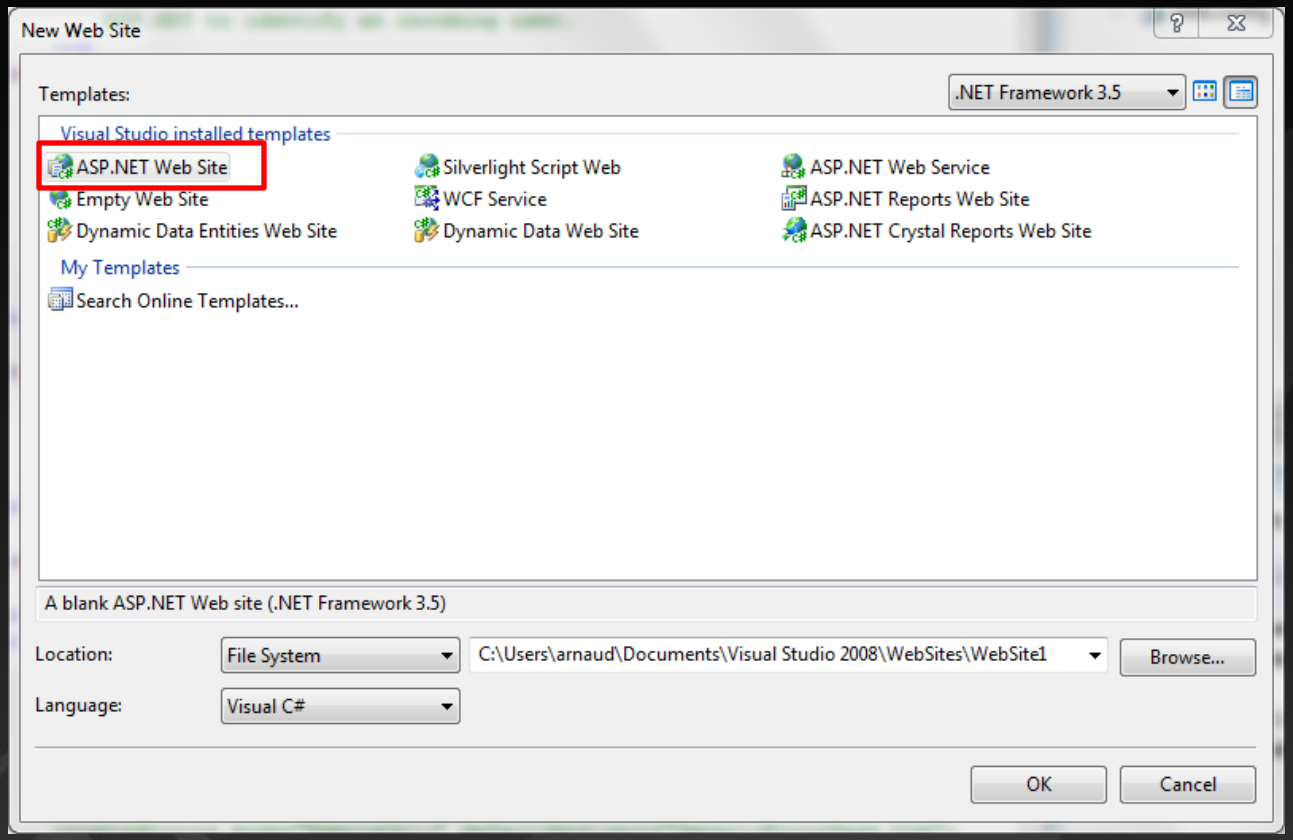
Profile

- Les différentes étapes :
  - Configurer le provider
  - Définir le profil d'un utilisateur
  - Rendre unique l'identification d'un user
  - Sauvegarder et récupérer les informations
  - Reconnaître un utilisateur

Profile

- De base, il existe un provider par défaut.

PROFILE


New Web Site
Templates:
.NET Framework 3.5
Visual Studio installed templates
ASP.NET Web Site
Empty Web Site
Dynamic Data Entities Web Site
Silverlight Script Web
WCF Service
Dynamic Data Web Site
ASP.NET Web Service
ASP.NET Reports Web Site
ASP.NET Crystal Reports Web Site
My Templates
Search Online Templates...
A blank ASP.NET Web site (.NET Framework 3.5)
Location:
File System
C:\Users\arnaud\Documents\Visual Studio 2008\WebSites\WebSite1
Browse...
Language:
Visual C#
OK
Cancel

PROFILE

Il faut compléter le fichier web.config, dans la section system.web

```
<anonymousIdentification enabled="true"/>
<profile>
  <properties>
      <add name="name" allowAnonymous="true"/>
      <add name="firstname" allowAnonymous="true"/>
  </properties>
</profile>
```

PROFILE

Pour l'utiliser, dans le code il suffit juste :

```
public partial class _Default : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        Label1.Text = Profile.name;
        Label2.Text = Profile.firstname;
    }
    protected void Button1_Click(object sender, EventArgs e)
    {

        Profile.name = "user";
        Profile.firstname = "user11";
    }
}
```

AUTHENTIFICATION

Authentification

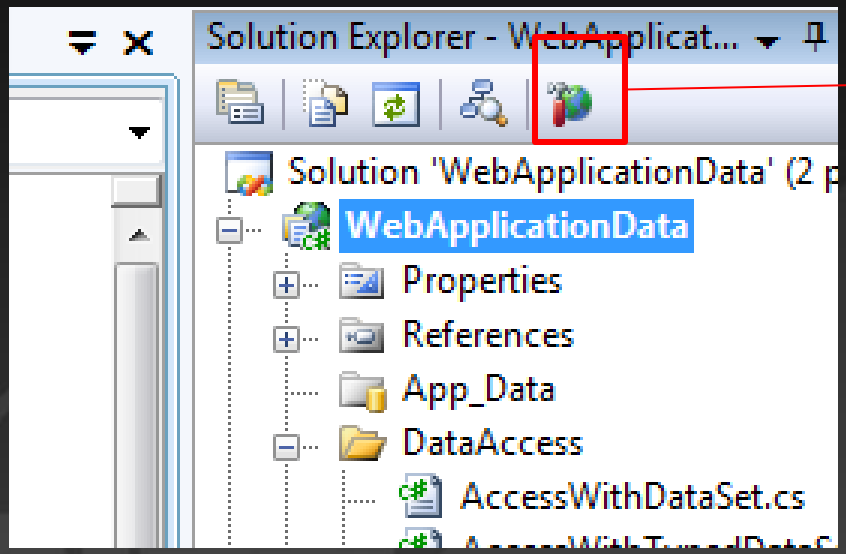
- Dès que l'on touche aux applications d'entreprises, généralement nous devons avoir un contrôle des données en back office.
  - Impose d'avoir des administrateurs
  - Des rôles
  - Des utilisateurs qui puissent se connecter

Authentification

- Pour cela ASP.NET nous offre plusieurs mécanismes tels que :
  - Les membership
  - Des wizards pour les configurations automatiques
  - Et des classes permettant de faire des contrôles personnalisés

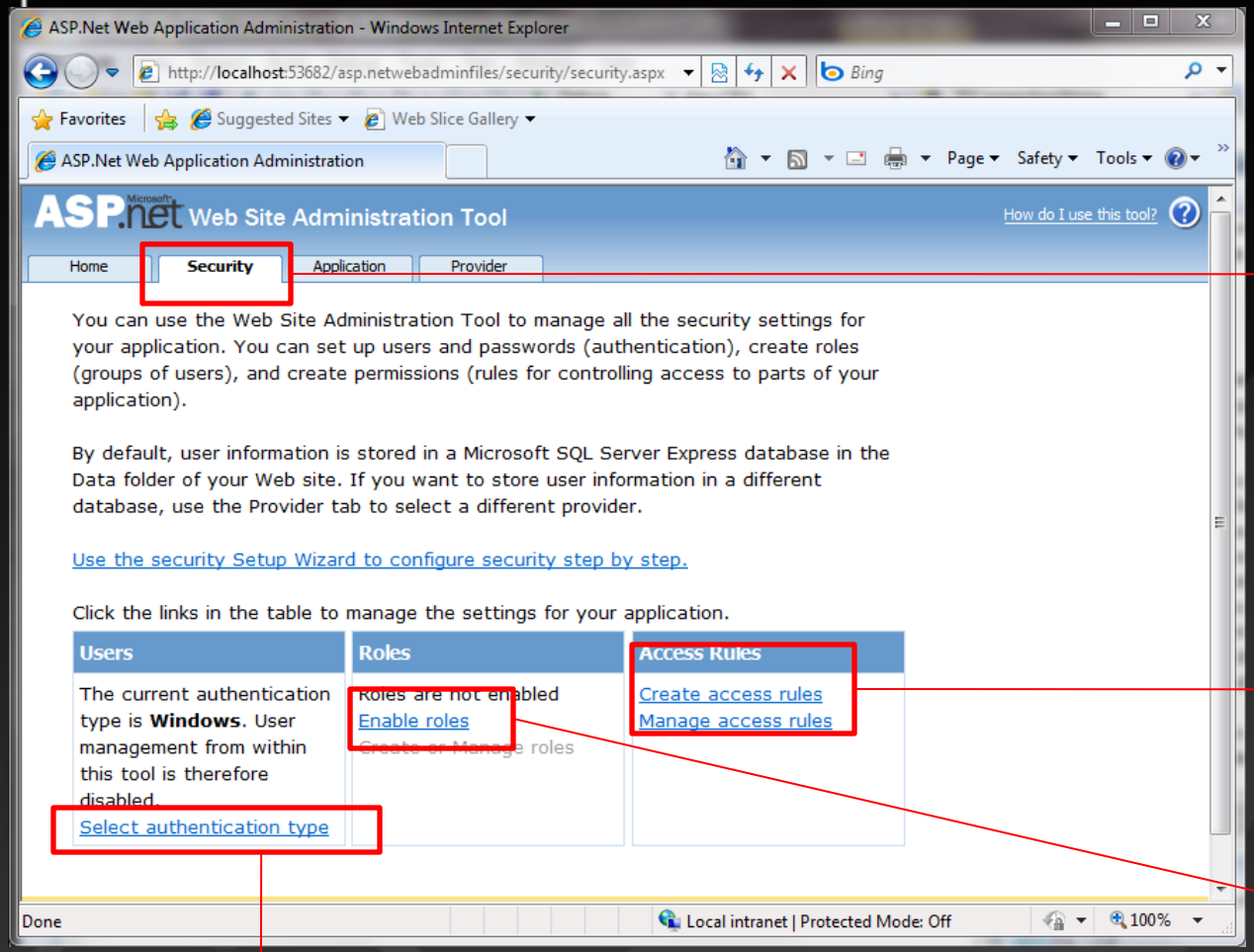
AUTHENTIFICATION

- De manière automatique :
  - Il faut sélectionner votre application web, puis cliquer sur asp.net configuration



Asp.net configuration

AUTHENTIFICATION



Onglet permettant de gérer les users, et les rôles

Pour chaque rôle on détermine les accès

Permet la gestion des rôles

Permet de sélectionner soit authentification windows, soit par formulaire

Authentification

- Cet outil est nommé WSAT pour WebSite Administration Tools. Il s'appuie sur une base de données SQL server qui est installée (si vous l'avez choisie) automatiquement lors de l'installation du framework.
- Cependant nous pouvons passer par notre propre système de stockage.

AUTHENTIFICATION

Il n'y a rien de magique, mis à part les users qui peuvent être stockés en base de données, tout se passe au niveau web.config de votre web application.

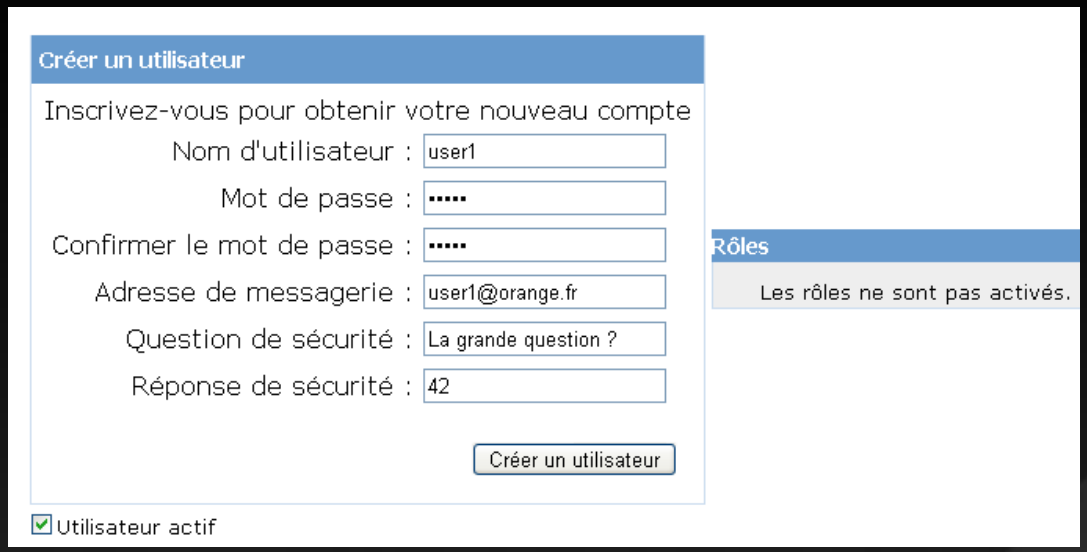
```
<system.web>
  [...]
  <authentication mode="Windows" />
  [...]
</system.web>
```

Gestion du mode d'authentification
Windows / Form /None …

```
<system.web>
  [...]
  <roleManager enabled="true" />
  [...]
</system.web>
```

Autorise la gestion des rôles d'asp.net

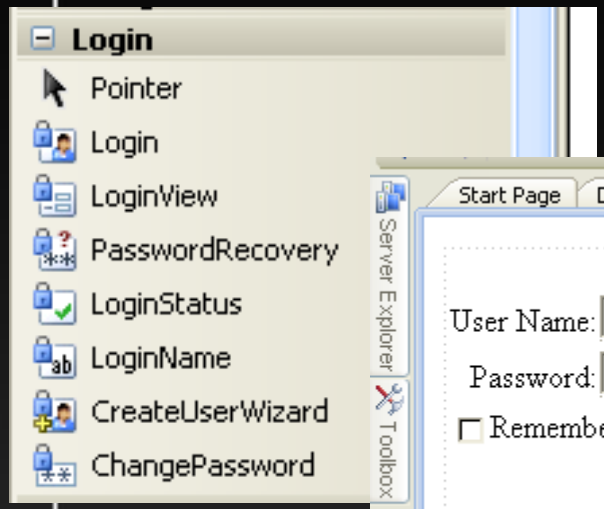
AUTHENTIFICATION

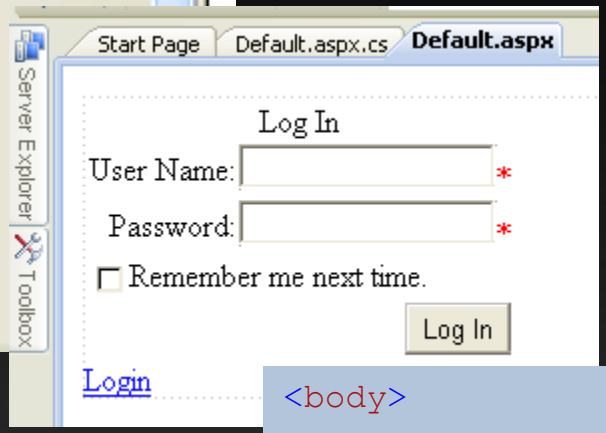


- Création d'un utilisateur dans l'interface d'administration

AUTHENTIFICATION

- Comment l'utiliser ?
  - - De façon très simple, il faut juste mettre les composants dont on a besoins.






```
<body>
    <form runat="server" >

    <asp:Login ID="Login1" runat="server">
    </asp:Login>
    <asp:LoginStatus ID="LoginStatus1" runat="server" />

    </form>
</body>
```

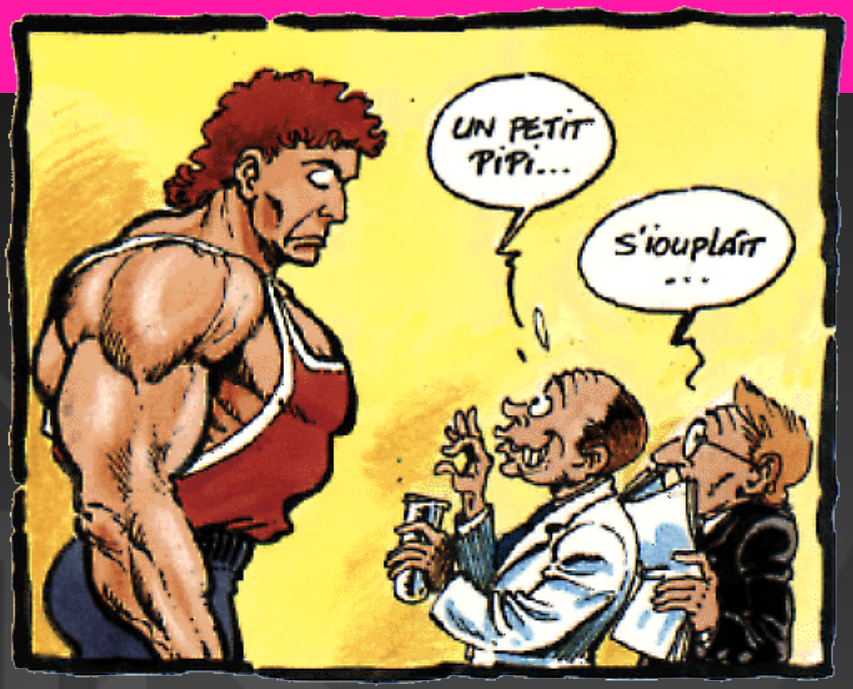
AUTHENTIFICATION

Selon le contexte on peut être amené à rediriger l'utilisateur directement sur une page de login. Pour cela le framework nous aide encore :

```
<authentication mode="Forms" >
  <forms loginUrl="login.aspx"/>
</authentication>
```

Cette section dans le web.config, permettra lors de l'accès d'un utilisateur à une ressource pour laquelle il n'est pas authentifié d'être redirigé automatiquement vers la page.

AUTORISATION



Autorisation

- Être authentifié est la 1ere étape mais encore faut-il que cela serve à quelque chose, c'est là que les autorisations sont importantes. Les autorisations définissent les accès pour les personnes authentifiées.

- Comment ça marche ?
  - Toujours dans le web.config

Autorisation

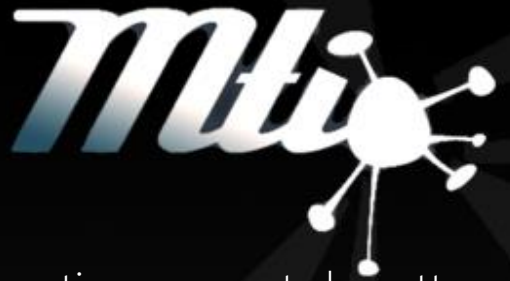
Les autorisations sont à définir dans le system.web du fichier de configuration.

```
<authorization>
   <deny users="?"/>
</authorization>
```

→ Permet de définir les autorisations pour le site en entier.

Généralement dans un site, il y a toujours une partie publique et seule la partie d'administration doit être protégée.
Pour protéger plus finement, il va falloir remonter les instructions plus haut dans le web.config

AUTORISATION



Dans la section configuration

```
<location path="">
  <system.web>
    <authorization>
      <deny users="?"/>
    </authorization>
  </system.web>
</location>
```

Le nœud location permet de setter l'attribut path qui détermine les dossiers à protéger.
« » => la racine du site
« Admin » => ~/Admin/

Ce nœud permet de définir, la politique pour ce répertoire.
Ce nœud peut contenir soit un a nœud deny soit allow

Les attributs peuvent être users et roles. Pour users les valeurs sont ? (anonymous user), * (tous les users), nom (nom d'un user.) séparés par des virgules s'il y en a plusieurs. Pour roles il faut rentrer le rôle paramétré pour l'application.

Autorisation

- Actuellement nous avons un système de protection efficace. Le seul problème actuellement est le fait que les users et rôle sont stockées dans une base de données, dont nous n'avons pas la maîtrise. On peut cependant mettre des users et des rôles directement dans le web.config.

PERSONNALISATION

Personnalisation

- Effectivement généralement un site d'entreprise possède déjà son propre référentiel de données avec ses utilisateurs et ses autorisations, le tout centralisé.
- Il faut donc construire un membership qui se basera sur notre base de données.

Personnalisation

- Les customs membership providers :

```
namespace WebApplicationSecurity
{
    public class CustomMemberShip : MembershipProvider
    {
        facultatif

        public override bool ValidateUser(string username, string password)
        {
            if (username == "user1" && password == "user11*")
            {
                return true;
            }
            else
            {
                return false;
            }
        }
    }
}
```

Personnalisation

Il faut maintenant indiquer d'utiliser notre classe plutôt que le provider par défaut.
Dans le fichier de config section : system.web

```
<membership defaultProvider="CustomMembershipProvider"
   userIsOnlineTimeWindow="15">
  <providers>
    <!--supprime toutes les références qui peuvent être déjà
   enregistrées-->
    <clear/>
    <add name="CustomMembershipProvider"
      type="WebApplicationSecurity.CustomMemberShip"
      applicationName="/"
    />
  </providers>
</membership>
```

Namespace + nom de la classe

Et voilà, le tour est joué, maintenant les formulaires de login passeront par notre classe.

Personnalisation

- De la même manière, nous pouvons également réaliser la même chose avec les roleproviders.

```
namespace WebApplicationSecurity
{
    public class CustomsRoles : RoleProvider
    {
        facultatif

        public override string[] GetRolesForUser(string username)
        {
            if (username == "user1")
            {
                return new string[] { "Admin", "SuperAdmin" };
            }
            else
            {
                return new string[] {"" };
            }
        }
    }
}
```

Personnalisation

Il faut également modifier le web.config dans la section system.web

```
<roleManager enabled="true"
   defaultProvider="CustomRoleProvider">
   <providers>
     <clear/>
     <add name="CustomRoleProvider"
   type="WebApplicationSecurity.CustomsRoles"/>
   </providers>
</roleManager>
```

POURQUOI FAIRE DE LA SÉCURITÉ ?

Pourquoi faire de la sécurité ?

- Attention, ce qui suit ne doit en aucun cas être tenté sur un autre site.
- Les conseils suivants ne sont donnés qu'à titre pédagogique.

Article 323-1
(Ordonnance n° 2000-916 du 19 septembre 2000 art. 3 Journal Officiel du 22 septembre 2000 en vigueur le 1er janvier 2002) (Loi n° 2004-575 du 21 juin 2004 art. 45 I Journal Officiel du 22 juin 2004)

Le fait d'accéder ou de se maintenir, frauduleusement, dans tout ou partie d'un système de traitement automatisé de données est puni de deux ans d'emprisonnement et de 30000 euros d'amende.

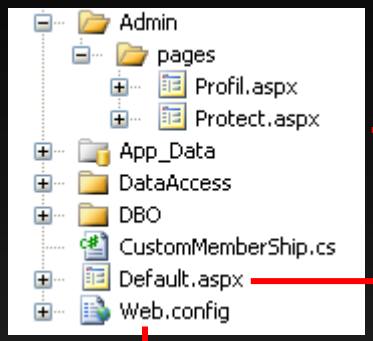
Lorsqu'il en est résulté soit la suppression ou la modification de données contenues dans le système, soit une altération du fonctionnement de ce système, la peine est de trois ans d'emprisonnement et de 45000 euros d'amende.

Pourquoi faire de la sécurité ?

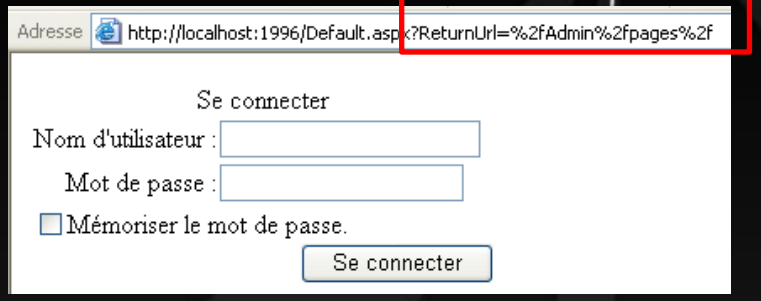
- Dans ce site de nombreuses failles de sécurités ont été insérées :
  - Faille de type SQL Injection
  - Faille de type ViewState

Pourquoi faire de la sécurité ?

- Structure du site :

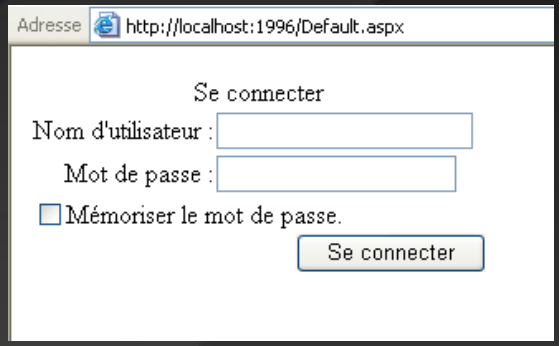


Pages protégées par mot de passe



Accès à la page sans authentification

Login Accessible par tout le monde



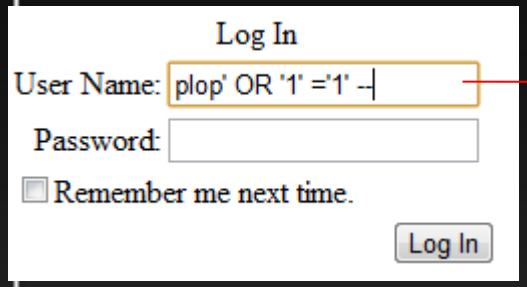
Impossible à télécharger bloqué par IIS

Base de données :

Name	Firstname	City	Password
lemettre	arnaud	Paris	arnaud
Comptehack	h4ck3r	toulouse	h@ck
Hoquet	guy	Paris	yves

Pourquoi faire de la sécurité ?

- 1er étape : Accéder aux pages protégées.
  - Il faut ajouter des caractères spéciaux de type : ‘ & | pour détecter les zones de sql injection
  - Puis entrer la requête pour passer



Sql injection dans le champs login

```
public static bool ValidUser(string username, string password)
{
    DataSet s = new DataSet();
    using (SqlConnection connection = new SqlConnection(@"Data Source=.\SQLEXPRESS;AttachD
    {
        SqlDataAdapter a = new SqlDataAdapter("select * from t_users where Name = '"+
                                              username +"' and Password = '"+
                                              password +"';",
                                              connection);

        a.Fill(s);
    }

    return s.Tables.Count != 0 ? s.Tables[0].Rows.Count != 0 : false;
}
```

La requête sera donc la suivante
select * from t_users where
Name = ‘plop’ OR ‘1’ = ‘1’ --

Met tout le reste en commentaire

Pourquoi faire de la sécurité ?

- 2ème étape : des informations peuvent se cacher dans les pages.

Personne inscrite dans la même ville que moi :

Nom
lemettre
Hoquet

Ensemble des Personnes inscrite sur le site :

Nom	Prénom
lemettre	arnaud
Comptehack	h4ck3r
Hoquet	guy

Reload Mon profil

Le viewState est encodé en 64 bits, en le décryptant on peut obtenir des informations

Décodeur 64 bits : http://www.motobit.com/util/base64-decoder-encoder.asp

Source data from the Base64 string:

```
      <Name>lemettre</Name>
      <Firstname>arnaud</Firstname>
      <Password>arnaud</Password>
    </Users>
    <Users diffgr:id="Users2" msdata:rowOrder="1"
diffgr:hasChanges="inserted">
      <Name>Comptehack</Name>
      <Firstname>h4ck3r</Firstname>
      <Password>h4ck3r</Password>
    </Users>
    <Users diffgr:id="Users3" msdata:rowOrder="2"
diffgr:hasChanges="inserted">
```

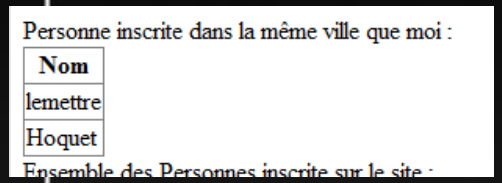
Type (or copy-paste) some text to a textbox bellow. The text can be Base64 string to decode or any string to encode to a Base64.

```
/wEPDwULLTE0MzUxMTc2NDAPFgIeB19fZGF0YTLkDgABAAAA/////wEAAAAAAAAADAIAAABOU3lzd
GVtLkRhdGEsIFZlcnNpb249NC4wLjAuMCwgQ3VsdHVyZT1uZXV0cmFsLCBQdWJsaWNLZXlUb2tlbj
1iNzdhNWM1NjE5MzRlMDg5BQEAAAAVU3lzdGVtLkRhdGEuRGF0YVRhYmxlAwAAABlEYXRhVGFibGU
uUmVtb3RpbmdWZXJzaW9uCVhtbFNjaGVtYQtYbWxEaWZmR3JhbQMBAQ5TeXN0ZW0uVmVyc2lvbIA
AAAJAwAAAAYEAAAAvgY8P3htbCB2ZXJzaW9uPSIxLjAiIGVuY29kaW5nPSJ1dGYtMTYiPz4NCjx4c
zpzY2hlbWEgeG1sbnM9IiIgeG1sbnM6eHM9Imh0dHA6Ly93d3cudzMub3JnLzIwMDEvWE1MU2NoZW
1hIiB4bWxuczptc2RhdGE9InVybjpzY2hlbWFzLW1pY3Jvc29mdC1jb206eG1sLW1zZGF0YSI+DQo
gIDx4czplbGVtZW50IG5hbWU9IlVzZXJzIj4NCiAgICA8eHM6eHM6c2VxdWVuY2U+DQogICAgICA8
eHM6c2VxdWVuY2U+DQogICAgICAgIDx4czplbGVtZW50IG5hbWU9Ik5hbWUiIHR5cGU9InhzOnN0c
mluZyIgbXNkYXRhOnRhcmdldE5hbWVzcGFjZT0iIiBtaW5PY2N1cnM9IjAiIC8+DQogICAgICAgID
x4czplbGVtZW50IG5hbWU9IkZpcnN0bmFtZSIgdHlwZT0ieHM6c3RyaW5nIiBtc2RhdGE6dGFyZ2V
0TmFtZXNwYWNlPSIiIG1pbk9jY3Vycz0iMCIgLz4NCiAgICAgICAgPHhzOmVsZW1lbnQgbmFtZT0i
```

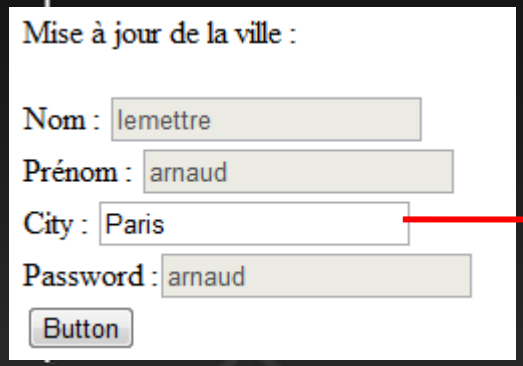

Les mots de passe ici sont apparents
Bien sur c'est pour l'exemple, dans la vraie vie c'est plutôt des Id que l'on pourra trouver pour s'en servir dans d'autres attaques sur le site.

Pourquoi faire de la sécurité ?

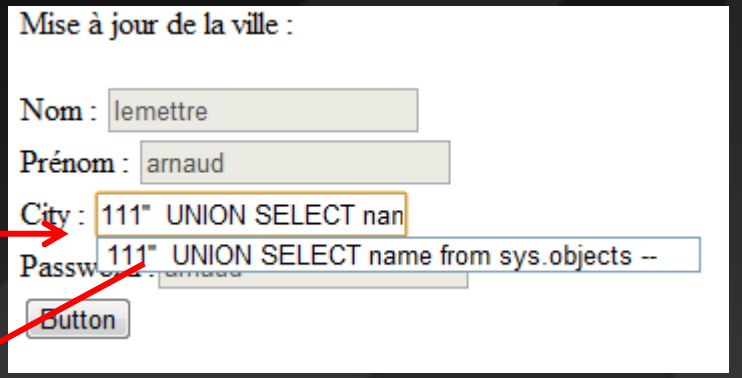
- 3ème étape : Admettons que nous avons un compte avec peu d'accréditation ;nous allons détourner les requêtes pour afficher des informations de la base de données.



Cette liste se construit selon la ville de l'utilisateur
Dans notre page profil on peut modifier notre ville.



Sql injection



Mettre deux simple quote afin de pouvoir en enregistrer une en base

Un pré-requis à une attaque par UNION est de déterminer le nombre de champs que l'on peut remonter ici 1

Pourquoi faire de la sécurité ?

- 3ème étape : Résultat de l'attaque
- Lorsque l'on revient sur la page protect.aspx, c'est cette requête qui est exécutée :

```
using (SqlConnection connection = new SqlConnection(@"Data Source=.\SQLEXPRESS;AttachI
{
    SqlDataAdapter a = new SqlDataAdapter("select  Name from t_users where City = '"
                                          +user.City +"'", connection);

    a.Fill(s);
}

if (s.Tables.Count != 0 && s.Tables[0].Rows.Count != 0)
```

Avec la propriété city modifiée :

select Name from t_users where City = '111' UNION SELECT name from sys.objects --'

Il faut que la 1ère condition ne renvoie aucun résultat

De cette manière seul ce résultat sera remonté, le name de sys.objects va remonter toutes les tables de la base

Désactive la dernière simple quote

Pourquoi faire de la sécurité ?

- Lorsque l'on retourne sur la page Protect.aspx nous avons maintenant à la place des noms , le nom des tables de la base :

Personne inscrite dans la même ville que moi :

Nom
EventNotificationErrorsQueue
filestream_tombstone_2073058421
sysxmlprops
T_Users

Ensemble des Personnes inscrite sur le site :

Nom	Prénom
lemettre	arnaud
Comptehack	h4ck3r
Hoquet	guy

Reload Mon profil

Maintenant que l'on a le nom des tables on peut faire des updates sur la base grâce aux failles trouvées précédemment

Pourquoi faire de la sécurité ?

- En conclusion :
  - En apparence, on peut voir qu'un site est protégé cependant en creusant un peu on voit apparaître de nombreuses failles.
  - C'est pourquoi il faut être très vigilant lorsque l'on code, afin de ne rien laisser au hasard.
  - Avec l'utilisation des nouveaux concept en .Net (linq – Entity Framework, …) ce genre de problème ne doit plus arriver

QUESTIONS ?