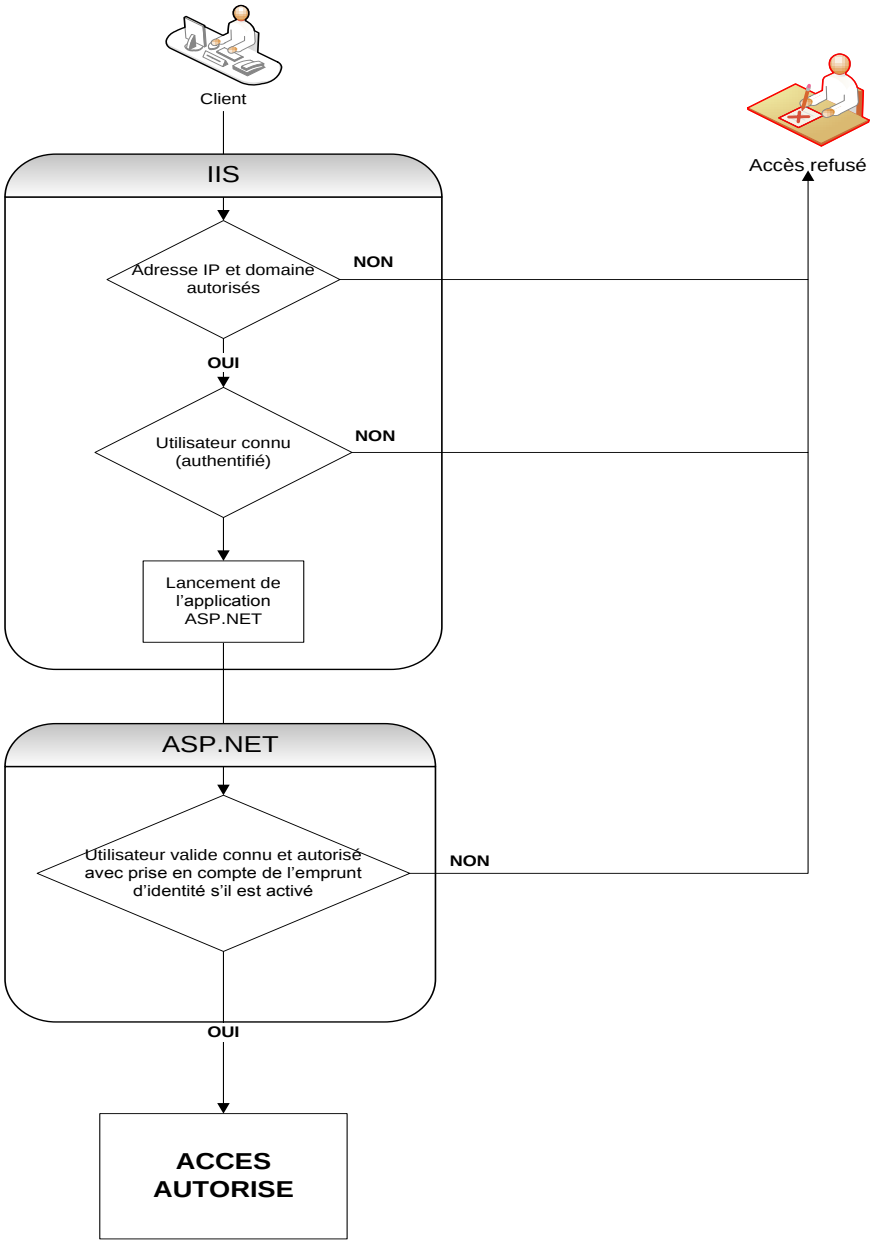
Sécurisation d'une application ASP.NET

1- Authentification

L'authentification est un processus essentiel à la sécurisation d'une application internet. Ce processus permet d'authentifier l'entité à l'origine d'une requête et de l'identifier auprès d'une autorité. L'autorisation détermine quelles sont les ressources accessibles à l'entité étant à l'origine de la requête.

IIS et ASP.NET mettent à la disposition des développeurs plusieurs systèmes d'authentification. Nous ferons abstraction dans cet article de la configuration de l'authentification sous IIS (Authentification anonyme, Authentification de base, Authentification Digest, Authentification Windows).
Pour des informations détaillées n'hésitez pas à me contacter: arthur[at]supinfo.com.

Principe de fonctionnement utilisant l'interaction IIS / Framework .NET :



ASP.NET permet l'authentification grâce à des fournisseurs d'authentification qui contiennent le code nécessaire pour authentifier et identifier un utilisateur.
ASP.NET supporte quatre types de fournisseurs d'authentification:
forms, windows (par défaut), passport, none (authentification personnalisée).

Les différents types d'authentification ASP.NET

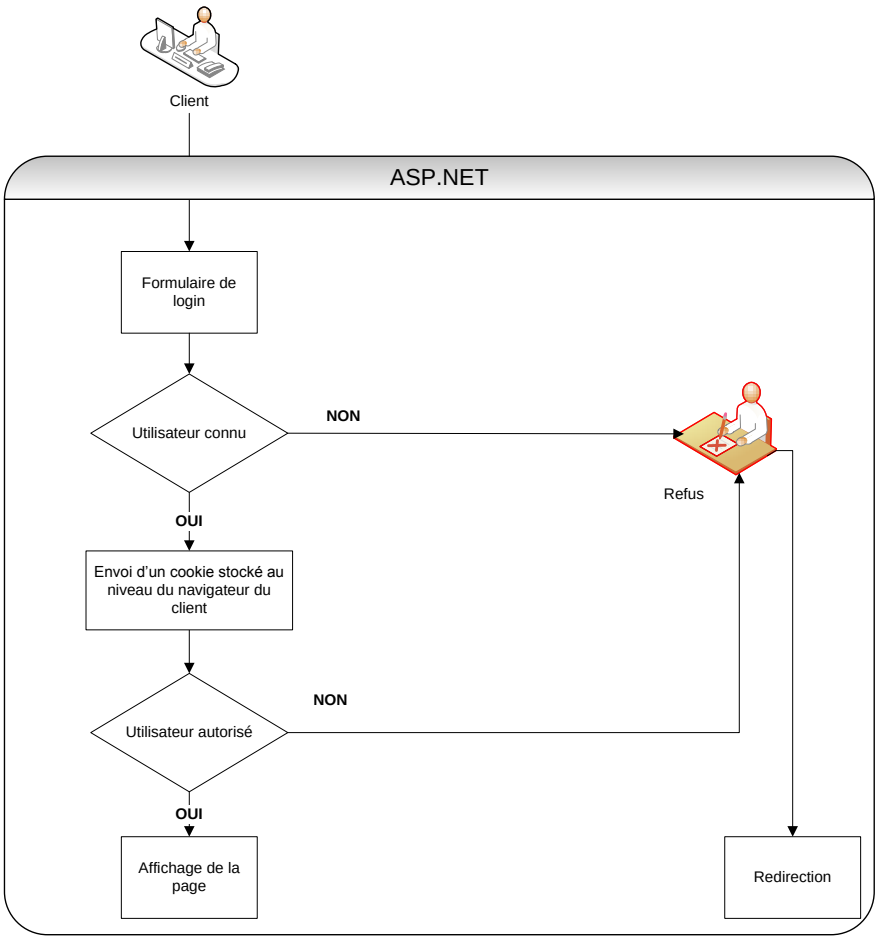
A- Authentification par Windows

Ce principe d'authentification permet de traiter l'utilisateur fourni par IIS comme un utilisateur authentifié dans une application ASP.NET. Cette méthode d'authentification affecte certes la valeur de la propriété User d'un WindowsIdentity fourni par IIS, mais il ne modifie en aucun cas l'identité Windows fournie par le Système. Or, les vérifications des autorisations d'accès se basent sur cette identité Windows. Il est donc important de calquer l'identité Windows ASP.NET sur l'identité Windows IIS en activant l'emprunt d'identité (identity avec l'attribut impersonate actif).

Configuration :

```
<system.web>
    <authentication mode = "Windows" />
    <identity impersonate = "true" />
</system.web>
```

B- Authentification par Forms

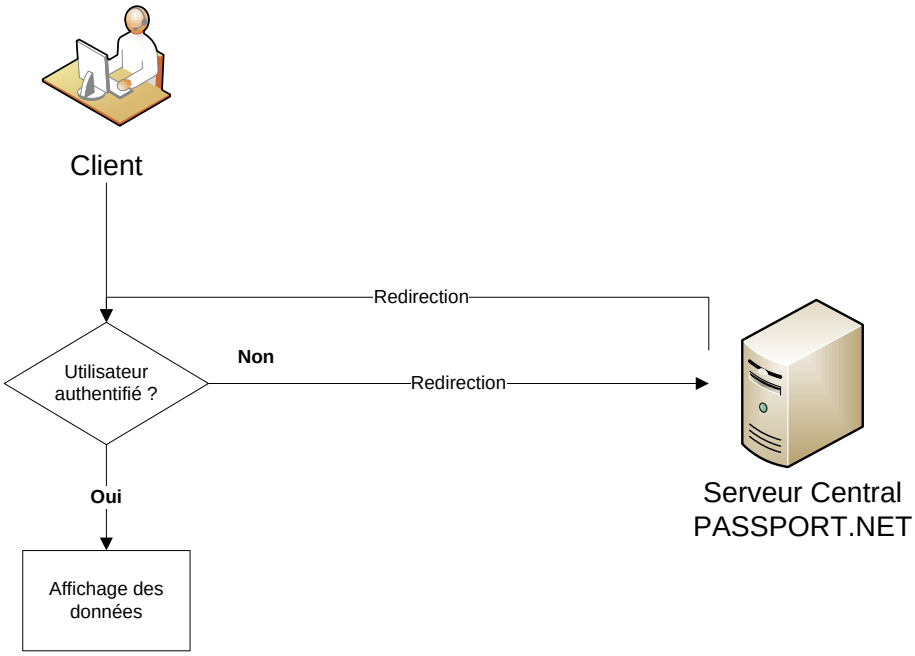


L'authentification par Forms est la méthode la plus couramment utilisée sur internet. Cette méthode d'authentification permet de gérer l'identification d'un utilisateur par un formulaire web. L'application ASP.NET stock les informations utilisateur dans un cookie. Ce type d'authentification permet de stocker ses propres informations client. Ce type d'authentification peut être mis en place très rapidement en utilisant le framework membership intégré au Framework .NET

Configuration :

```
<system.web>
   <authentication mode = "Forms">
      <forms name=".ASPXFORMSAUTH" loginUrl="Login.aspx" />
   </authentication>
</system.web>
```

C- Authentification par Passport



Cette méthode d'authentification est un service qui permet aux utilisateurs de créer un compte unique sécurisé permettant l'accès à tous les services et sites Web .NET Passport. Ces sites reposent sur le serveur .NET Passport qui s'occupe de l'authentification. Ainsi vous pouvez établir une connexion sécurisée (SSL) entre votre site et le serveur central .NET Passport. Tout en permettant aux utilisateurs de conserver leur compte Passport, d'un site à l'autre. La communication ne s'effectue pas en temps réel entre le serveur central (.Net Passport) et votre site Web. Les échanges d'informations son gérés au niveau du navigateur du client en utilisant une redirection HTTP.

Configuration :

```
<system.web>
   <authentication mode = "Passport" />
</system.web>
```

2- Gestion simplifiée des comptes utilisateur

Le Framework .Net met à disposition des développeurs des outils de gestion utilisateur très puissants qui permettent de gérer la création de compte, l'identification et la réinitialisation de mot de passe en quelques clicks.

A- Framework Membership

Membership est un Framework gérant l'abstraction des données utilisateur. Cet outil permet d'interagir avec la base de données utilisateur en fournissant une classe et un ensemble de méthodes essentielles pour la gestion de comptes. Pour récupérer les données utilisateur, la classe Membership fait appel à des providers.

Le Framework .NET inclut un provider (SqlMembershipProvider) qui permet de stocker les informations utilisateur dans une base de données (par défaut configuré pour SQL Server) ainsi qu'un provider (ActiveDirectoryMembershipProvider) qui permet de stocker les informations utilisateur sur un serveur Active Directory. Un provider personnalisé peut par ailleurs être mis en place afin qu'il communique avec une autre source de données. Par défaut, le provider utilisé par Membership est "SqlMembershipProvider".

Configuration :

```
<membership defaultProvider = "Prov">
  <providers>
    <add name=" Prov" type="System.Web.Security.SqlMemberShipProvider, System.Web,
Version=2.0.0.0, Culture=neutral" connectionStringName="MySqlServer" />
  </providers>
</membership>
```

B- Contrôles prédéfinis

Un contrôle serveur est une entité issue de System.Web.UI.Control permettant de générer du code HTML. Des contrôles serveur spécifiques à gestion de comptes utilisateur sont fournis par le framework.net (LoginView, Login, PasswordRecovery, …).
Ces contrôles permettent d'automatiser les formulaires de gestion de comptes utilisateur.

3- Rôles et Profiles utilisateur

A- Rôles

Les rôles permettent de définir des types d'utilisateur, ainsi il devient possible de lier l'accès à certaines parties d'un site à certains types d'utilisateur.
L'utilisation de rôles peut se faire dans le cadre d'une authentification forms ou d'une authentification windows. L'authentification windows est donc très utile dans le cadre d'un intranet, alors qu'une authentification par forms s'avère beaucoup plus cohérante dans le cadre d'un site internet.

Configuration:

- Les providers

```
<system.web>
  <membership defaultProvider="RoleProv" enable="true" cacheRolesInCookie = "true">
    <roleManager enabled="true" />
    <providers>
                <add name="RoleProv" type="System.Web.Security.SqlRoleProvider,
System.Web, Version=2.0.0.0, Culture=neutral" connectionStringName="MySqlServer" />
    </providers>
  </membership>
</system.web>
```

- Les restrictions générales

```
<location path="edition.aspx">
  <system.web>
    <authorization>
      <allow roles="editeur" />
      <deny users="?"  />
    </authorization>
  </system.web>
</location>
```

- Affectation et test d'un rôle

```
if(!Roles.IsUserInRole("editeur"))
    Roles.AddUserToRole(User.Identity.Name,"editeur");
```

B- Profiles utilisateur

Les profiles utilisateur peuvent s'avérer très utiles dans le cadre de la personnalisation d'un site web (design). En effet les profiles permettent de stocker dans un cookie des informations spécifiques à un utilisateur pour un rôle donné.

Configuration :

Profile anonyme:

```
<anonymousIdentification enabled="true" cookieless="AutoDetect" />
```

Propriétés d'un profile:

```
<system.web>
  <profile enabled="true">
    <properties>
      <add name="couleur" type="System.String" />
    </properties>
  </profile>
</system.web>
```

Gestion des profiles dans le code-Behind:

```
ProfileCommon currentProfile = Profile.GetProfile(e.AnonymousID);
 if (currentProfile != null)
      Profile.color = currentProfile.Color;
```

Arthur Tarlay
Laboratoire des technologies WEB
SUPINFO AQUITAINE
IS1 – Promotion 2008