Manipuler l’authentification biométrique de Windows Seven

Une grande majorité des ordinateurs portables vendus actuellement dispose d’un grand nombre de fonctionnalités (lecteur de SmartCard, prise HDMI, etc.) qui sont, pour la plupart, souvent inutilisées.

Cependant, avec l’arrivée de Windows Seven, l’une de ces fonctionnalités prend tout son sens: **l’authentification biométrique**. En effet, jusqu’à maitenant, cette authentification devait passer par des logiciels tiers, la plupart du temps développés les constructeurs.

Fort heureusement, Windows Seven dispose d’une nouvelle API (<http://msdn.microsoft.com/en-us/library/dd401509(VS.85).aspx)> permettant de programmer/manipuler les lecteurs biométriques de votre ordinateur. Ainsi, il devient possible, avec un peu de PInvoke, de créer un petit wrapper de l’API managé.

Pour vous simplifier la vie, j’ai créé ce petit wrapper managé. Pour le moment, seule la méthode **WinBioIdentifyWithCallback** est implémentée (mais il est facilement possible de l’enrichir). Cette méthode permet de vérifier, de façon asynchrone, que l’utilisateur qui vient d’utiliser le lecteur biométrique est effectivement un utilisateur reconnu:

**IMAGE 1.png**

**IMAGE 2.png**

Dans le cas d’un utilisateur non reconnu, l’application reçoit un évènement indiquant la cause du problème:

**IMAGE 3.png**

Au niveau de l’utilisation, c’est tout ce qu’il y a de plus simple:

static void Main(string[] args)

{

var biometric = new BiometricWrapper();

biometric.IdentifySuccess += new BiometricWrapper.OnIdentifySuccessHandler(biometric\_IdentifySuccess);

biometric.IdentifyFailed += new BiometricWrapper.OnIdentifyFailedHandler(biometric\_IdentifyFailed);

biometric.IdentifyAsynchronous();

Console.ReadLine();

}

static void biometric\_IdentifySuccess(AuthenticationSuccessEventArgs args)

{

User selectedUser = WMISearcher.FindUser(args.User);

Console.WriteLine("User SID: " + selectedUser.SID);

Console.WriteLine("User Name: " + selectedUser.Name);

Console.WriteLine("User Description: " + selectedUser.Description);

Console.WriteLine("User Status: " + selectedUser.Status);

}

static void biometric\_IdentifyFailed(AuthenticationFailedEventArgs args)

{

Console.WriteLine("Identification failed: " + args.Error);

}

Il est à noter que ce wrapper repose sur 3 DLLs:

* **Biometric.Wrapper.dll**: Il s’agit de la DLL managée faisant appel aux méthodes non managées
* **Interop.WinBio.dll**: C’est la DLL qui contient la définition des méthodes non managées (autrement dit le code PInvoke). Plutôt que d’écrire moi-même ce code, j’ai préféré le générer automatiquement au moyen de l’outil **PInvoker** (<http://www.pinvoker.com>). C’est rapide, efficace et en cas de problèmes, l’auteur de l’outil est très réactif donc n’hésitez pas à y jeter un oeil !
* **PInvoker.Marshal.dll**: DLL générée par l’outil PInvoker et qui contient la définition des types utilisés dans Interop.WinBio.dll

Pour télécharger le wrapper (en version minimaliste), c’est par ici: <http://morpheus.developpez.com/blog/BiometricWrapper/BiometricWrapper.zip>

Thomas LEBRUN

Consultant / Formateur

<http://blogs.developpeur.org/tom/>

Microsoft MVP Client Application Development (WPF / Silverlight)