CAS
Code Access Security

Implémenter CAS

Présentation

- CAS: (Sécurité d'accès du code) mécanisme de sécurité qui permet de mesurer le degré de confiance que l'on peut accorder à un assemblage en vérifiant sa source et son intégrité et permet de définir les permissions accordées à un assemblage.
- Eléments du mécanisme CAS
  - Preuve
  - Permission
- Le mécanisme est utilisé par le CLR dans les cas suivants:
  - Les permissions sont accordées à un assemblage lors de son chargement.
  - Lorsque le code demande l'exécution d'une opération qui nécessite des privilèges alors le CLR vérifie les permissions de l'assemblage

Preuve

- Une preuve est une information extraite à partir d'un assemblage est qui prouve sa provenance et son intégrité.
- Types de preuves (System.Security.Policy):
  - ApplicationDirectory: Fournit le répertoire de l'application comme preuve
  - Url: Site Url de téléchargement de l'application
  - Publisher: Assemblage signé avec Authenticode X.509 v3
  - Hash: code de hachage de l'application généré au moment de la compilation de l'assemblage.
  - Zone Internet (Enumération: System.Security.SecurityZone)
    - Internet
    - Untrasted: Sites sensibles
    - Trusted: Sites de confiance
    - Intranet
    - MyComputer.
    - NoZone

- Une instance de la classe System.Security.Policy.Evidence contient une collection pour stocker les preuves fournies par l'application.
- Exemple: Affichage des preuves de l'application en cours

```
Evidence evidence= AppDomain.CurrentDomain.Evidence;
IEnumerator preuves=evidence.GetHostEnumerator();
 while( preuves.MoveNext())
      textBox1.Text += preuves.Current + "\n";
```

- Résultat:

```
<System.Security.Policy.Url version="1">
<Url>file:///D:/Cours/cours_secu/CAS/bin/Debug/CAS.EXE</Url>
</System.Security.Policy.Url>

<System.Security.Policy.Zone version="1">
<Zone>MyComputer</Zone>
</System.Security.Policy.Zone>

<System.Security.Policy.Hash version="2">
<hash algorithm="SHA1"
value="BEA9D1FD4146AD911EC14394212518FCFC9FEDA8"/>
<hash algorithm="SHA256"
value="F492F4AD1D974B03B3B3746F8E8AC1D521BCC3596EA6E2DDDE15875A63BF218B"/>
<hash algorithm="MD5"
value="57FB2E6C4D566CC6D9338F7E26105A88"/>
</System.Security.Policy.Hash>
```

- Exercice: afficher les types de preuves fournies par les assemblages du domaine d'application courant, éléments de la solutions:
  - AppDomain.CurrentDomain.GetAssemblies(): retourne les assemblages du domaine d'application courant de type Assembly[]
  - Un Assembly possède les propriétés FullName et Evidence.
  - La méthode GetType() retourne le type d'une classe sous la forme d'une chaîne de caractères.

Les permissions

- Le framework .Net distingue quatre types de permissions:
  - Permissions standards
  - Permissions d'identité
  - Méta-permissions
  - Permissions propriétaires

Permissions standards

- Classe de base System.Security.CodeAccessPermission, contient les méthodes pour s'assurer qu'on a une permission (IsUnrestricted) , de demander une permission (Demand) , de refuser une permission (Deny)
  - System.Security.Permissions.**EnvironmentPermission**
  - System.Security.Permissions.**FileDialogPermission**
  - System.Security.Permissions.**FileIOPermission**
  - System.Security.Permissions.**IsolatedStoragePermission**
  - System.Security.Permissions.**ReflectionPermission**
  - System.Security.Permissions.**RegistryPermission**
  - System.Security.Permissions.**UIPermission**
  - System.Security.Permissions.**DataProtectionPermission**
  - System.Security.Permissions.**KeyContainerPermission**

- System.Security.Permissions.**StorePermission**
- System.Security.Permissions.**SecurityPermission**
- System.Configuration.**UserSettingsPermission**
- System.Security.Permissions.**ResourcePermissionBase**
- System.Diagnostics.**EventLogPermission**
- System.Diagnostics.**PerformanceCounterPermission**
- System.DirectoryServices.**DirectoryServicesPermission**
- System.ServiceProcess.**ServiceControllerPermission**
- System.Net.**DnsPermission**
- System.Net.**SocketPermission**
- System.Net.**WebPermission**
- System.Net.NetworkInformation.**NetworkInformationPermission**
- System.Net.Mail.**SmtpPermission**
- System.Web.**AspNetHostingPermission**
- System.Messaging.**MessageQueuePermission**
- System.Drawing.Printing.**PrintingPermission**
- System.Data.Common.**DBDataPermission**
- System.Data.OleDb.**OleDbPermission**
- System.Data.SqlClient.**SqlClientPermission**
- System.Data.Odbc.**OdbcPermission**
- System.Data.OracleClient.**OraclePermission**
- System.Data.SqlClient.**SqlNotificationPermission**
- System.Transactions.**DistributedTransactionPermission**

Permissions d'identité

- Pour chaque preuve, le CLR accorde à l'assemblage une permission d'identité.
  - System.Security.Permissions.PublisherIdentityPermission
  - System.Security.Permissions.UrlIdentityPermission
  - System.Security.Permissions.ZoneIdentityPermission
- Les permissions d'identité dérivent aussi de la classe CodeAccessPermission
- Les permissions d'identité sont utilisées uniquement dans le cadre des vérifications.

Méta-permissions (ou permissions de sécurité)

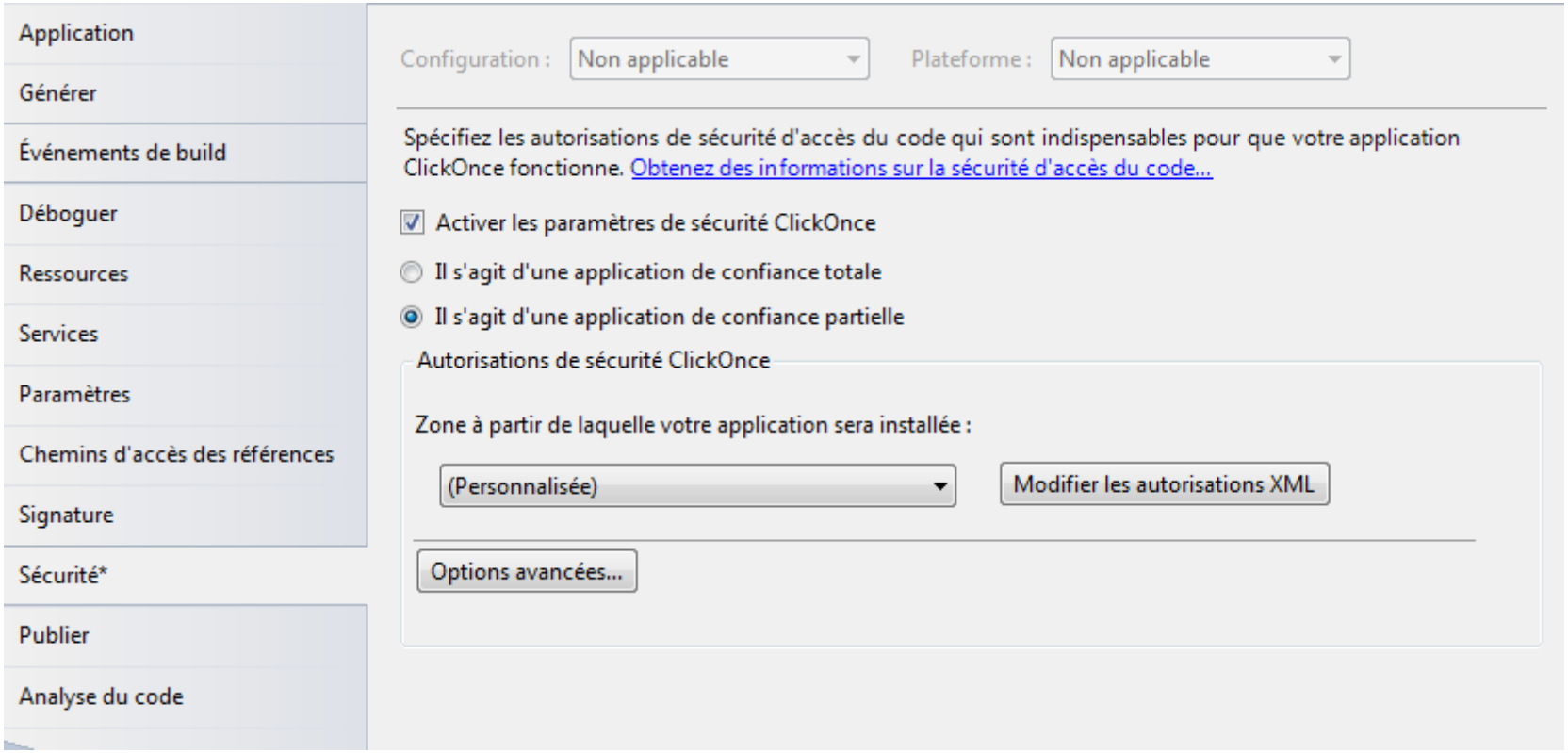
- Permissions allouées au gestionnaire de sécurité définies par l'énumérations SecurityPermissionFlag, valeurs possibles:
  - UnmanagedCode: permission d'exécuter du code non managé (non géré)
  - ControlEvidence:permission de fournir des preuves à partir d'un assemblage.
  - SkipVerification: exécuté du code non protégé

Exemple : vérifier la disponibilité d'une permission

```
try
        {
            System.Security.Permissions.FileIOPermission perm =
            new System.Security.Permissions.FileIOPermission(
            System.Security.Permissions.FileIOPermissionAccess.AllAccess,
            @"C:\\fichier.txt");
            perm.Demand();
            Console.WriteLine("Permission accordée");
        }
catch (Exception exp)
        {
            Console.WriteLine(exp.Message);
        }
```

Appliquer une Stratégie CAS

- Activer les paramètres de sécurité ClickOnce
- Choisir "Confiance partielle"
- Choisir l'une des zones par défaut (internet ou intranet) ou bien choisir "personnalisée" et définir les permissions dans le nœud *ApplicationRequestMinimum du* fichier xml app.manifest.xml.



Générer des permissions au format XML.

```
System.Security.Permissions.FileIOPermission permission= new
System.Security.Permissions.FileIOPermission(PermissionState.Unrestricted);
System.Security.SecurityElement element;
element = permission.ToXml();
System.IO.StreamWriter writer = new
System.IO.StreamWriter("D:\\permissions.xml");
writer.Write(element.ToString());
writer.Flush();
writer.Close();
```

- Ajouter le code xml généré dans le nœud *ApplicationRequestMinimum du fichier* app.manifest.xml

```
<IPermission class="System.Security.Permissions.FileIOPermission,
mscorlib, Version=4.0.0.0, Culture=neutral,
PublicKeyToken=b77a5c561934e089"
version="1"
Unrestricted="true"/>
```

Demander la permission UAC

- Configurer l'élément requestedExecutionLevel du nœud requestedPrivileges (pour l' UAC, l'application utilise par défaut les mêmes permissions que l'utilisateur de l'application 'asInvoker')
  - <requestedExecutionLevel level="asInvoker" uiAccess="false" />

  Pour demander les privilèges de l'administrateur:
  - <requestedExecutionLevel level="requireAdministrator" uiAccess="false" />

  Pour demander le niveau le plus élevé possible
  - <requestedExecutionLevel level="highestAvailable" uiAccess="false" />