

Signature numérique de script PowerShell

par [Laurent Dardenne \(Contributions\)](#)

Date de publication : 21/09/2007

Dernière mise à jour : 26/09/2007

PowerShell permet l'exécution de tâches répétitives au sein de scripts mais pour des raisons de sécurité leurs exécution nécessitent une vérification basée sur une signature numérique d'un éditeur approuvé et ce pour tous les scripts et fichiers de configuration, y compris les scripts que vous écrivez sur l'ordinateur local. Ce tutoriel vous propose de mettre en place une signature personnelle auto-approuvée afin d'exécuter vos scripts dans un environnement protégé.

- 1 - Public concerné
- 2 - Avertissements
- 3 - Signature et stratégies d'exécution
- 4 - Créer une autorité de certification auto-approuvé
- 5 - Créer un certificat
- 6 - Outils de gestion des certificats
 - 6-1 - Aperçus de la gestion des certificats avec MMC
- 7 - Signer un script PowerShell
- 8 - Protection du certificat
- 9 - Vérification de l'exécutable PowerShell
- 10 - Politiques
- 11 - Glossaire
- 12 - Liens concernant la signature de code


1 - Public concerné



Testé sous le Framework PowerShell 1.0 Fr, .NET 2.0 et XP sp2.

Pré requis : les outils du SDK .NET 2.0

2 - Avertissements

 *Ni l'auteur ni Developpez.com ne proposent de signature approuvée pour PowerShell, ils ne pourront donc être tenus pour responsable d'un usage malveillant du nom de la signature de script donné en exemple dans ce tutoriel.*

3 - Signature et stratégies d'exécution

PowerShell intègre des stratégies d'exécution qui assurent la sécurité de l'environnement de script en déterminant les conditions dans lesquelles PowerShell charge des fichiers de configuration et exécute des scripts, qui portent respectivement les extensions **.ps1xml** et **.ps1**

Les stratégies d'exécution sont :

Valeur	Description
Restricted	Valeur par défaut. Niveau de sécurité maximum, aucune exécution de script, accès interactif uniquement et charge uniquement les fichiers de configuration signés dont la signature a été approuvée.
AllSigned	Tous les fichiers .ps1 et .ps1xml doivent être signés numériquement. Si un fichier signé est exécuté, le shell vous demandera de confirmer si l'éditeur de la signature est autorisé ou non à exécuter ce fichier.
RemoteSigned	Seuls les fichiers .ps1 et .ps1xml provenant d'Internet doivent être signés numériquement. Si un fichier signé issue d'Internet est exécuté, le shell affichera un message d'avertissement demandant confirmation avant son exécution.
Unrestricted	Aucun fichier ne doit être signé. Si un fichier provient d'Internet, le shell affichera un message d'avertissement demandant confirmation avant son exécution. Pour supprimer ce message d'avertissement, faites, dans l'explorateur de Windows, un clic-droit sur le fichier, puis "propriétés" et enfin "débloquer".

Le niveau le plus restrictif (**Restricted**) n'offre que peu de sens à un shell pour automatiser des tâches dans un environnement professionnel.

Je vous propose de créer une signature numérique qui permettra l'exécution de scripts sous la stratégie d'exécution **AllSigned**, à savoir :

- les scripts peuvent être exécutés.
- Requier la signature numérique d'un éditeur approuvé sur tous les scripts et fichiers de configuration, y compris les scripts que vous écrivez sur l'ordinateur local.
- Vous demande confirmation avant d'exécuter des scripts provenant d'éditeurs approuvés.
- Risque d'exécuter des scripts signés, mais malveillants.

Vérifions la stratégie d'exécution actuelle :

```
PS C:\test\Get-ExecutionPolicy
Restricted
```

Créons notre script d'exemple :

```
PS C:\test\" # Script non signé affichant les fichiers du répertoire contenu">MonScript.ps1
PS C:\test\" # dans la variable d'environnement Windir">>MonScript.ps1
PS C:\test\"Dir $env:Windir">>MonScript.ps1
```

Puis exécutons-le, notez qu'on doit fournir un chemin pour l'exécuter :

```
PS C:\test\ .\MonScript.ps1
```

Si vous n'avez pas modifié la configuration de sécurité de PowerShell, son exécution affichera :

```
PS C:\test\ScriptsPowerShell> .\MonScript.ps1
Impossible de charger le fichier C:\Test\ScriptsPowerShell\MonScript.ps1.
Le fichier C:\Test\ScriptsPowerShell\MonScript.ps1 n'est pas signé numériquement.

Le script ne sera pas exécuté sur le système. Pour plus d'informations, consultez « get-help
about_signing »..
Au niveau de ligne : 1 Caractère : 13
+ .\MonScript.ps1 <<<<
```

Pour avoir le droit de l'exécuter il nous faut soit redéfinir la stratégie de sécurité en *unrestricted*, soit le signer numériquement, en ayant au préalable redéfini la stratégie de sécurité que l'on a choisi de mettre en #uvre:

```
PS C:\test\ScriptsPowerShell> Set-ExecutionPolicy AllSigned
```

 *L'instruction précédente nécessite les droits administrateur.*

Pour signer numériquement du code, ici sous forme de script, on utilise un certificat d'identité normalisé X509 (cf. [RFC2459](#)). Son rôle est d'assurer l'authenticité de l'émetteur et l'intégrité du contenu signé.

La technologie **authenticode** de Microsoft met à notre disposition des outils dédiés à la gestion des signatures numériques, ils sont répertoriés dans la rubrique **CryptoAPI** du SDK.


Ces outils sont classés en **2 catégories** :

- Signature de fichiers et contrôle de signatures (MakeCat.exe, SetReg.exe et SignTool.exe).
 - **MakeCat** Construit un fichier-catalogue non signé. Ce fichier-catalogue est employé ultérieurement pour comparer le hachage du fichier-catalogue aux hachages des différents dossiers, afin de vérifier que les fichiers originaux sont exempts de modifications.
 - **SetReg** Positionne les clefs de la base de registre qui commandent le procédé de vérification de certificat.
 - **SignTool** Signe numériquement des fichiers et confirme leur signature numérique.

- Création, visualisation et gestion des certificats (MakeCert.exe, Cert2SPC.exe, MakeCTL.exe et CertMgr.exe).
 - **MakeCert** Crée un certificat X.509.
 - **Cert2SPC** Crée un certificat d'éditeur de logiciel (SPC, Software Publisher's Certificate).
 - **MakeCTL** Crée une liste de certificat de confiance (CTL, Certificate Trust List). MakeCTL est également accessible via un assistant GUI.
 - **CertMgr** Gère les certificats, les CTL, et les listes de certificats révoqués (CRLs, Certificate Revocation Lists).

Pour consulter l'aide en ligne sur le sujet :

```
Get-Help about_signing | More
```

 L'outil en ligne de commande **pvk2pfx** permet de déplacer les informations de la clef publique et privée, contenue dans un fichier .spc, .cer, et .pvk, vers un fichier d'échange personnel d'information (.pfx). Un fichier PFX associe vos clefs publiques et privées dans un seul fichier.

Télécharger [pvk2pfx](#).

4 - Créer une autorité de certification auto-approuvé

Un certificat pour être approuvé doit être créé par une autorité de certification (CA, Certificate Authority). Elle vous assure, moyennant finance, qu'un certificat n'usurpe pas l'identité qu'il vous présente. Un certificat à une durée de vie limitée et peut être révoqué par le CA, à cette fin Authenticode maintient également en local une liste de révocation de certificat (CRL).

Vous trouverez [ici](#) les principales autorités de certification. Et ici un [aperçus](#) des documents à fournir en vu d'obtenir un certificat numérique certifié et approuvé.

Il reste possible de s'affranchir d'une autorité de certification en utilisant **les services de certification PKI**. Le serveur de certificats de clé publique est conçu pour les entreprises qui souhaitent délivrer des certificats de clé publique à leurs utilisateurs sans dépendre de services commerciaux d'autorités de certification (CA). Le site du **CNRS** propose quelques documents, plus organisationnel que technique, sur le sujet.


Voir aussi :  [PKI Enhancements in Windows XP Professional and Windows Server 2003](#)

En ce qui nous concerne nous utiliserons un *certificat auto-signé* gratuit, c'est à dire que nous ne disposerons pas d'une certification approuvée par un CA externe. Dans ce cas vous-même et l'ordinateur approuverez ce certificat, ce qui est suffisant pour un usage privé ou dans un contexte de développement.

On utilise l'outil **makecert**, disponible dans le SDK .NET. Créons donc notre propre CA en ligne de commande :

```
&"C:\Program Files\Microsoft Visual Studio 8\SDK\v2.0\Bin\makecert.exe" `
-n "CN=Laurent Dardenne autorité de certification racine locale" `
-a sha1 -eku 1.3.6.1.5.5.7.3.3 -r -sv LDAPowerShell.pvk LDAPowerShell.cer -ss Root -sr localMachine
```

Attention la commande précédente ne doit pas contenir de retour chariot, on utilise à cette fin le caractère apostrophe inverse (') accessible via la combinaison de touches Alt Gr + 7.

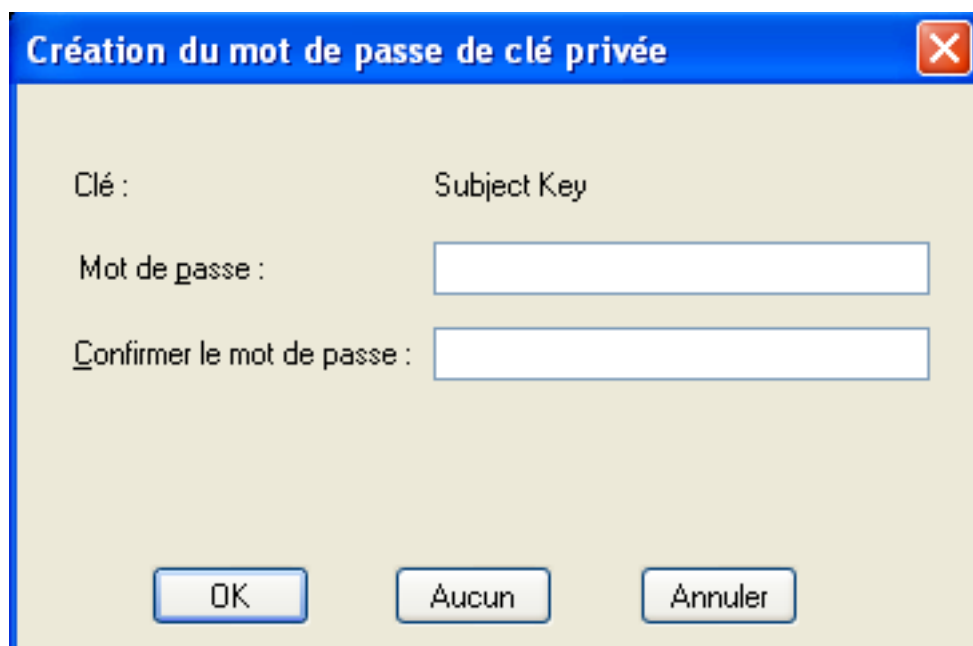
 Notez la présence du caractère **&** nécessaire, sous PowerShell, à l'exécution d'un programme contenant des espaces dans le chemin d'accès.

Cet appel est décomposé ainsi :

Paramètre	Signification
-n	Nom commun, c'est un identificateur unique respectant la norme X500. Ex: / C=CountryCode/O=Organization/ OU=OrganizationUnit/CN=CommonName.
-a	Indique l'algorithme de hachage pour l'empreinte.
-eku	Insert dans le certificat un OID précisant comment le certificat sera utilisé par une application. Ici


	il s'agit de la signature de code (cf. XCN_OID_PKIX_KP_CODE_SIGNING).
-r	Crée un certificat 'auto-signé'.
-sv	Emplacement du fichier de clé privée .pvk.
-ss	Nom du magasin de certificat où sera stocké le certificat créé.
-sr	Emplacement du magasin de certificat dans la registry, soit localMachine (HKLM) ou currentUser (HKCU).

L'exécution de makecert avec ces paramètres nous demande la saisie du mot de passe de la clé privée :



cette étape crée le fichier suivant dans le répertoire d'exécution :

LDPowerShell.pvk

 *Mémoriser le mot de passe que vous venez de saisir, il vous sera demandé pour d'autres opérations liées à ce certificat.*

Le fichier .pvk doit être supprimé lors d'une ré exécution de makecert sinon il ne sera pas régénéré.

la seconde étape nous demande la saisie du même mot de passe utilisé précédemment :

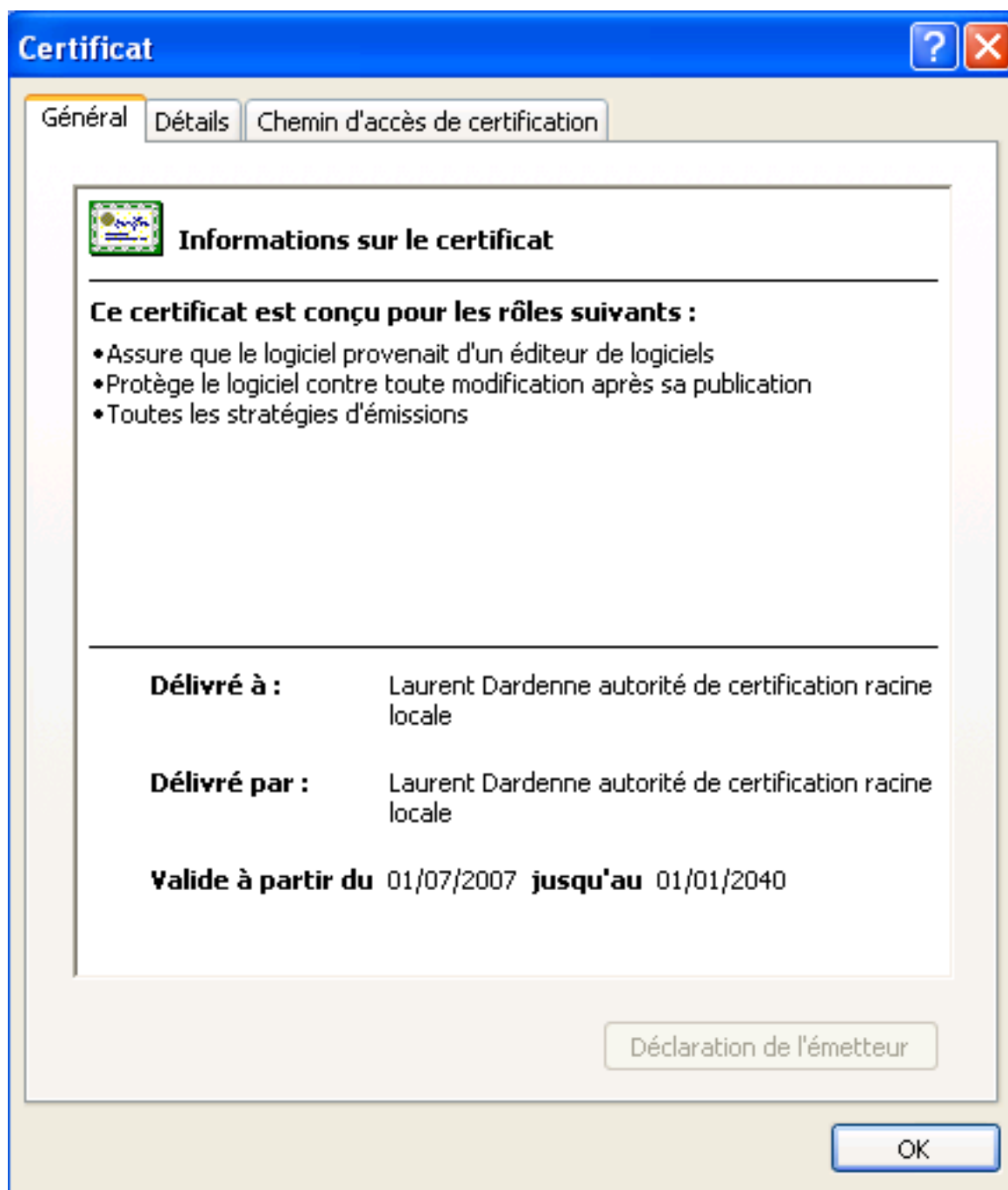


le fichier suivant est créé dans le répertoire d'exécution :

LDPowerShell.cer

 *Notez que le fichier .pvk est verrouillé durant l'exécution de ce programme.*

Le fichier .cer contient notre certificat, un double clic sous l'explorer nous affiche son contenu :



A l'aide de PowerShell recherchons dans les magasins l'inscription de notre certificat :

```
cd cert: # Se place dans le provider de certificat
dir -recurse|where {$_ -match "CN=Laurent"} # Recherche le nom d'après les paramètres indiqués à
makecert
c:
```

Le résultat nous indique que notre certificat se trouve dans les magasins d'autorités de certification racine pour l'utilisateur courant ainsi que pour la machine locale :

```

Répertoire : Microsoft.PowerShell.Security\Certificate::CurrentUser\Root

Thumbprint                               Subject
-----
445103E2111EC4472B72E96B4675B2796C4304CB  CN=Laurent Dardenne autorité de certification racine
locale

Répertoire : Microsoft.PowerShell.Security\Certificate::LocalMachine\Root

Thumbprint                               Subject
-----
445103E2111EC4472B72E96B4675B2796C4304CB  CN=Laurent Dardenne autorité de certification racine
locale
    
```

Ce certificat peut être visualisé à l'aide de MCC que nous verrons plus avant dans ce tutoriel :



5 - Créer un certificat

Une fois notre autorité de certification créée et installée il reste à créer le certificat de signature. On saisira le même mot de passe qu'utilisé précédemment :

```
&"C:\Program Files\Microsoft Visual Studio 8\SDK\v2.0\Bin\makecert.exe" -pe -n "CN=Laurent Dardenne
certificat pour PowerShell" `
-ss My -a sha1 -eku 1.3.6.1.5.5.7.3.3 -iv LDAPowerShell.pvk -ic LDAPowerShell.cer
```

Attention la commande précédente ne doit pas contenir de retour chariot, on utilise à cette fin le caractère apostrophe inverse (') accessible via la combinaison de touches Alt Gr + 7.

Cet appel est décomposé ainsi :

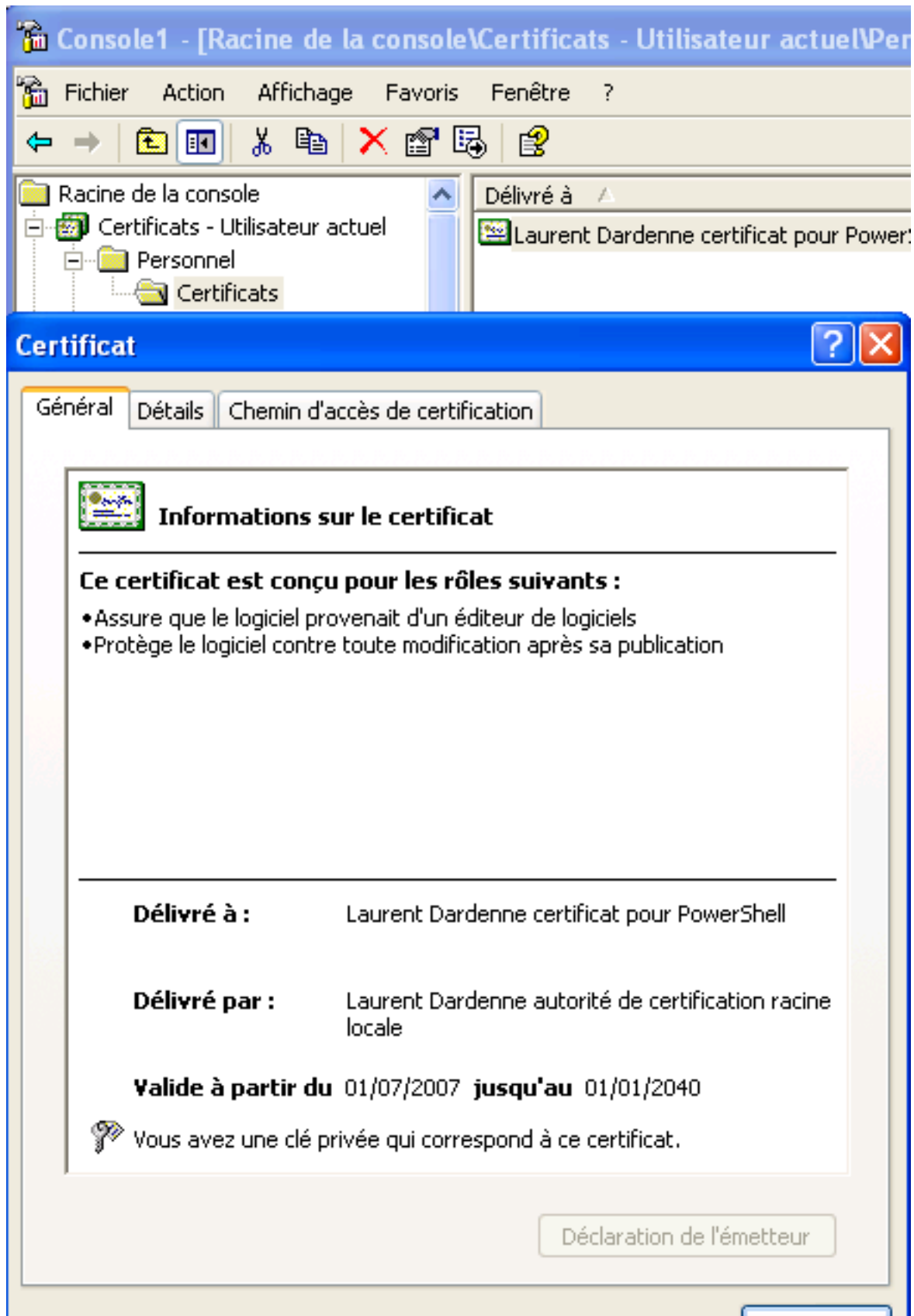
Paramètre	Signification
-pe	Marque la clé privée générée comme étant exportable. Cela permet l'inclusion de la clé privée dans le certificat.
-n	Nom commun, c'est un identificateur unique respectant la norme X500. Ex: / C=CountryCode/O=Organization/ OU=OrganizationUnit/CN=CommonName.
-ss	Nom du magasin de certificat où sera stocké le certificat créé.
-a	Indique l'algorithme de hachage pour l'empreinte.
-eku	Insère dans le certificat un OID précisant comment le certificat sera utilisé par une application. Ici il s'agit de la signature de code (cf. XCN_OID_PKIX_KP_CODE_SIGNING).
-iv	Spécifie le fichier de clé privée .pvk de l'émetteur.
-ic	Spécifie le fichier de certificat de l'émetteur.

Recherchons de nouveau dans les magasins de certificats :

```
Répertoire : Microsoft.PowerShell.Security\Certificate::CurrentUser\My
```

```
Thumbprint                               Subject
-----
D485E53BEA01F6D30F72F25D2511876D5CCCA67F CN=Laurent Dardenne certificat pour PowerShell
```

Ce certificat est inscrit directement dans le magasin personnel (My) :



6 - Outils de gestion des certificats

Pour visualiser le contenu des magasins de certificat nous disposons de plusieurs outils, qui sont :

- le gestionnaire de certificat, certmgr.exe,
- Internet Explorer (menu outils-options internet, onglet "contenu" zone 'certificats'),
- la console MMC (Mmc.exe),
- ou encore le provider de certificats de PowerShell (*cer:*).

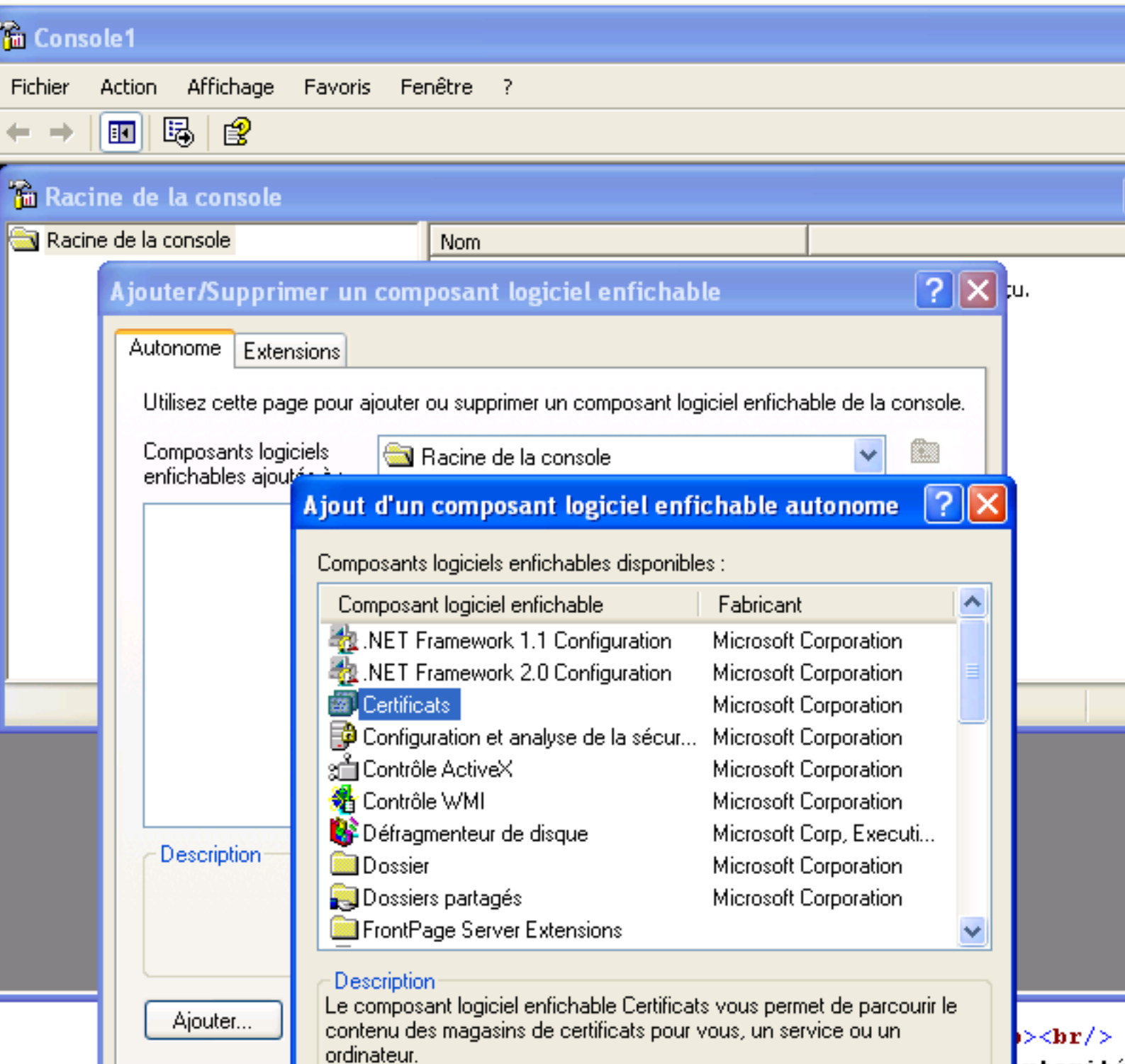
Aperçu de la console Certificates - Utilisateur actuel \ Autorités de certification racines de confiance \ Certificates

Fichage	Favoris	Fenêtre	?
Utilisateur actuel			
Autorité de certification racine			
Entreprise			
Délicré à	Délicré par	Date d'expiration	
Laurent Dardenne autorité de certification...	Laurent Dardenne autorité de...	01/01/2040	
Microsoft Authenticode(tm) Root Authority	Microsoft Authenticode(tm) R...	01/01/2000	
Microsoft Code Signing PCA	Microsoft Root Authority	25/09/2011	
Microsoft Corporation	Microsoft Code Signing PCA	06/04/2006	
Microsoft Root Authority	Microsoft Root Authority	21/12/2020	

Le cmdlet **Get-PSprovider** renvoi tous les providers disponibles.

6-1 - Aperçus de la gestion des certificats avec MMC

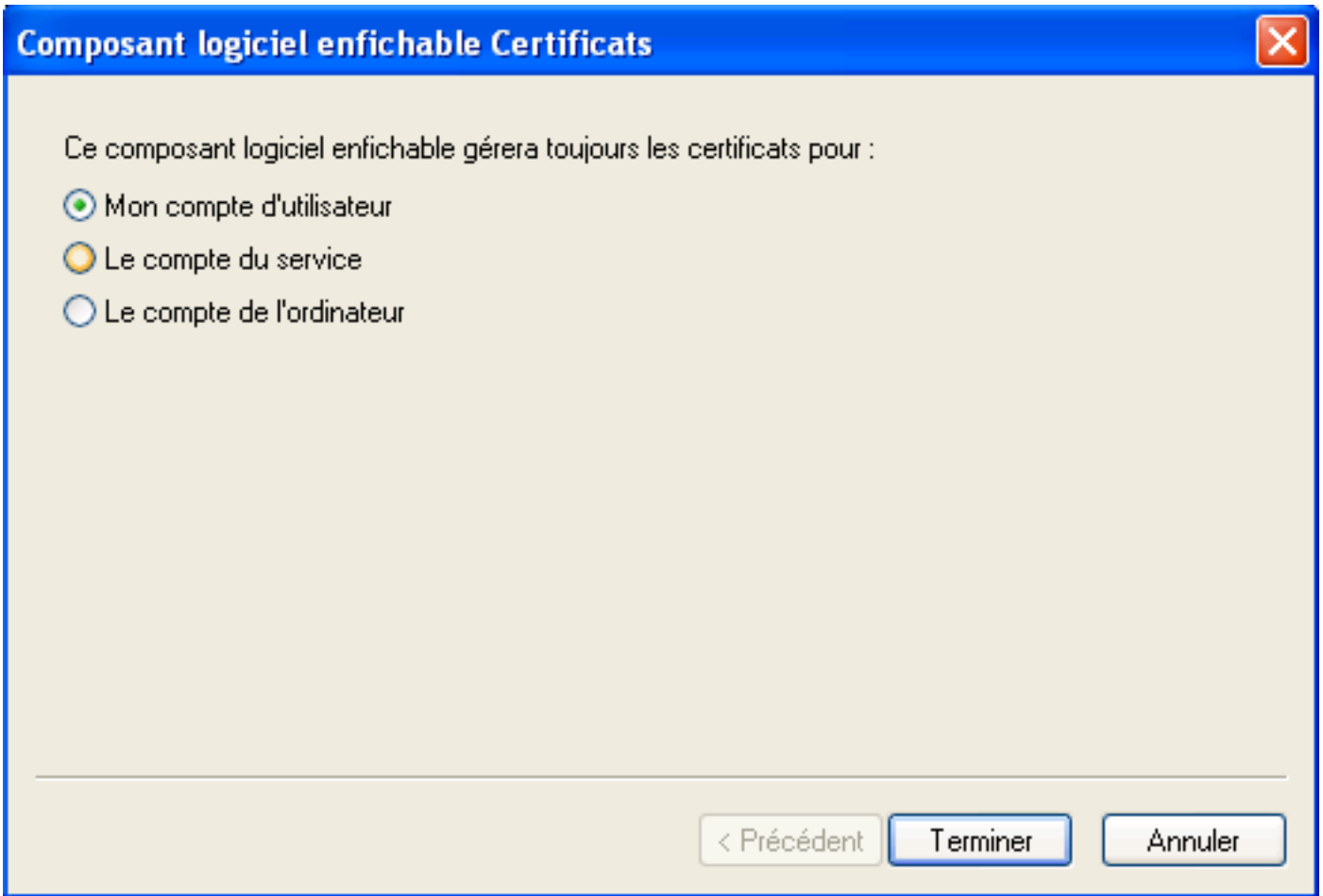
Exécutez la console d'administration, en saisissant *MMC.exe*. Dans le menu fichier sélectionnez "Ajouter/supprimer un composant logiciel enfichable..." :



Console mmc

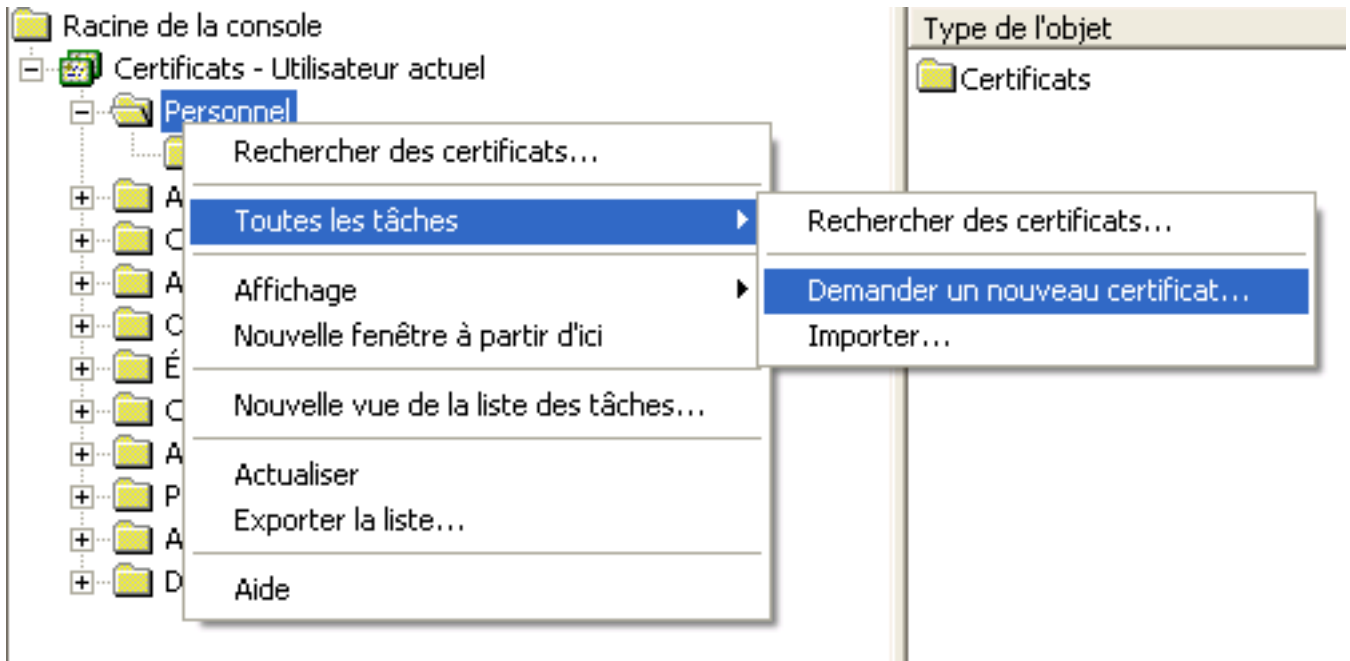
Cliquez sur le bouton "Ajouter", choisissez "Certificats" puis cliquez sur "Ajouter".

Sélectionnez l'option "Mon compte utilisateur" :



Cliquez sur le bouton "Terminer" puis sur "Fermer" et enfin sur "Ok".

Il est possible de demander un nouveau certificat en cliquant sur le menu contextuel du dossier nommé *Personnel*



Etant sur un ordinateur isolé la demande ne peut aboutir :



La même opération avec l'option "Le compte de l'ordinateur" est plus explicite :

Certificats (ordinateur local)

Personnel

Aucun élément à afficher dans cet aperçu

Assistant Demande de certificat



L'Assistant ne peut pas être démarré à cause d'une ou plusieurs des conditions suivantes :

- Il n'y a pas d'Autorités de certification (CA) approuvées disponibles.
- Vous n'avez pas l'autorisation de faire la demande de certificats à partir des autorités de certification suivantes.
- Les Autorités de certification disponibles émettent des certificats pour lesquels vous n'avez pas d'autorisation.

OK



Un serveur de signature au sein d'une **infrastructure PKI** est nécessaire pour utiliser ces fonctionnalités.

7 - Signer un script PowerShell

Utilisons le cmdlet approprié, à savoir **Set-AuthenticodeSignature**

```
$cert=@(Get-ChildItem cert:\Currentuser\My)[0]
Set-AuthenticodeSignature MonScript.ps1 $cert
```

L'affichage du résultat nous indique que l'opération a réussi :

SignerCertificate	Status	Path
----- D5855EBFCA607A562B7EE725E85AD7F83721A4CD	Valid	----- MonScript.ps1

le même résultat sera affiché par le cmdlet **Get-PfxCertificate** :

```
Get-PfxCertificate MonScript.ps1
```

La visualisation du contenu du script par **Get-Content MonScript.ps1** nous le confirme :

```
# script non signé affichant les fichiers du répertoire contenu dans la variable d'environnement
Windir
Dir $env:Windir

# SIG # Begin signature block
# MIIEBgYJKoZIhvcNAQcCoIIEXzCCBFsCAQExCzAJBgUrDgMCGgUAMGkGCisGAQQB
... (Ligne supprimée afin de réduire la taille de l'affichage)
# 5J2RT7kM7d41Y5nCbDd+GUa259i4sYvGUQhQZWUfGvLV9w==
# SIG # End signature block
```


On peut désormais exécuter ce script, les concepteurs de PowerShell ont préférés forcer la syntaxe suivante afin de réduire les risques d'exécution *à la volée*


```
.\MonScript.ps1
```

Lors de la première exécution PowerShell nous demande l'action à effectuer :

```
Voulez-vous exécuter le logiciel de cet éditeur non approuvé ?

Le fichier MonScript.ps1 est publié par CN=Laurent Dardenne certificat pour PowerShell et n'est pas
approuvé sur votre système.
N'exécutez que des scripts provenant d'éditeurs approuvés.
[M] Ne jamais exécuter [N] Ne pas exécuter [O] Exécuter une fois [T] Toujours exécuter [?] Aide
(la valeur par défaut est « N ») :
```

 *Il va s'en dire qu'il est nécessaire de s'assurer du contenu d'un script, téléchargé ou recopié en local, avant son exécution.*

 *Si on répond **O** (Exécuter une fois) ce message de confirmation sera de nouveau affiché lors de la prochaine exécution d'un script possédant cette signature.*

Si on modifie le fichier d'origine ou si on reconstruit un script (test.ps1) en y copiant la signature, on obtient le message d'erreur suivant nous informant d'une falsification du script courant :

```
Impossible de charger le fichier Test.ps1. Le contenu du fichier Test.ps1 peut avoir été falsifié,
car le hachage du fichier ne correspond pas à celui qui figure dans la signature numérique.
Le script ne sera pas exécuté sur le système.
```

```
Pour plus d'informations, consultez « get-help about_signing »..
Au niveau de ligne : 1 Caractère : 79
Test.ps1 <<<<
```

Si on déplace ou copie le certificat, utilisé pour signer ces scripts, dans le magasin de *certificats non autorisés*, PowerShell affiche le message suivant :

```
Le fichier MonScript.ps1 est publié par CN=Laurent Dardenne certificat pour PowerShell.
Cet éditeur est explicitement désapprouvé sur votre système.
Ce script ne sera pas exécuté sur le système.
```

```
Pour plus d'informations, consultez « get-help about_signing ».
Au niveau de ligne : 1 Caractère : 15
+ .\MonScript.ps1 <<<<
```


Bien évidemment si on le retire de la liste de révocation, son exécution est de nouveau possible.

Comme les certificats ont une durée de vie limitée, voir les paramètres **-d** et **-e** de makecert, il est possible de se retrouver, lors de l'exécution d'un script, devant le message d'erreur suivant :

```
Impossible de charger le fichier C:\temp\MonScript.ps1. Un certificat requis n'est pas dans sa
période de
validité selon la vérification par rapport à l'horloge système en cours ou le tampon daté dans le
fichier signé.
```

```
Au niveau de ligne : 1 Caractère : 25
+ C:\temp\tst\MonScript.ps1 <<<<
```

On peut donc désormais modifier le fichier de configuration de PowerShell puisqu'à l'origine il est signé par Microsoft et ne peut être modifié sans effectuer une nouvelle opération de certification.

 *Il reste possible d'utiliser le programme **SignTools.exe**, fournis avec le SDK, pour effectuer la même opération de signature.*

En mode assistant ou en ligne de commande.


8 - Protection du certificat

Des programmes malveillants peuvent signer des scripts en utilisant votre certificat privé afin d'empêcher ce type d'opération vous devez exporter votre certificat dans un fichier .pfx afin de le protéger *via* une demande de mot de passe.

Vous trouverez la marche à suivre dans le document nommé *about_signing.txt* livré avec PowerShell.

 *On peut également protéger les scripts via **EFS**.*

9 - Vérification de l'exécutable PowerShell

L'exécutable *PowerShell.exe* pouvant être lui même la cible de détournement, sa vérification se fera à l'aide l'outil *signtool.exe* au travers d'un fichier catalogue comme indiqué sur le  [blog de l'équipe de PowerShell](#).

 *Le catalogue pour la version localisée Fr se nomme KB926140.cat.*

```
&"C:\Program Files\Microsoft Visual Studio 8\SDK\v2.0\Bin\signtool.exe" verify /c KB926140.CAT  
powershell.exe  
Successfully verified: powershell.exe
```

10 - Politiques

Microsoft met à disposition un outil de gestion de politiques pour PowerShell permettant de gérer les stratégies d'exécution.

Windows PowerShell

Turn on Script Execution

Afficher les [Propriétés](#)

Configuration requise :
At least Microsoft Windows XP or Windows Server 2003 family

Description :
This settings lets you configure the script execution policy, controlling what scripts are allowed to run.

If you enable this setting, the scripts selected in the drop down list will be allowed to run.

The "Allow only signed scripts" setting allows script to execute only if they are signed by a trusted publisher.

The "Allow local scripts and remote signed scripts" setting allows any local scrips to run; scripts that originate from the Internet must be signed by a trusted publisher.

The "Allow all scripts" setting allows all scripts to run.

If you disable this setting, no scripts are allowed to run.

Note: This setting exists under both "Computer Configuration" and "User Configuration" in the group policy editor. The "Computer Configuration" has precedence over "User Configuration."

If this policy setting is not configured or disabled, the setting reverts to a per-machine preference setting; the default if that is not configured is "No scripts allowed."

Paramètre	État
Turn on Script Execution	Activé

Propriétés de Turn on Script Execution

Paramètre Expliquer

Turn on Script Execution

Non configuré

Activé

Désactivé


Execution Policy: Allow only signed scripts

Pris en charge sur : At least Microsoft Windows XP or Window

Paramètre précédent Paramètre suivant

OK An

Télécharger [Administrative Templates for Windows PowerShell](#)

 Les fichiers de **profile** (`profile.ps1`) ainsi que les fichiers ***.ps1xml** doivent être impérativement signés et posséder des droits d'accès restreint.

Note:

Il reste un dernier problème lors du déploiement. Etant donné que la première exécution d'un script demande l'approbation manuelle du certificat, l'automatisation de cette opération s'avère difficile, pour le moment et faute de temps je n'ai pas trouvé de solution.

L'utilisation des API de Cryptographie peut être une piste...

11 - Glossaire

Certificat

Fichier signé numériquement qui contient les informations d'identification d'un utilisateur ou d'un serveur et sa clé publique, servant à vérifier son identité et établir un lien sécurisé. Les certificats peuvent contenir différents types de données. Par exemple, un certificat X.509 inclut le format du certificat, le numéro de série du certificat, l'algorithme employé pour signer le certificat, le nom du CA qui a publié le certificat, la clé nommée et publique de l'entité demandant le certificat, et la signature de CA.

CA certificate (CA)

Autorité de certification délivrant des certificats électroniques et se chargeant de l'authentification des clients et serveurs utilisant ces certificats.

Certificate store

Typiquement c'est un lieu de stockage permanent où on entrepose des certificats, des listes de révocation de certificat (CRLs), et les listes de confiance de certificat (CTLs).

Certificate revocation list (CRL)

Un document maintenu et édité par une autorité de certification (CA) qui liste les certificats invalides publiés par un CA.

Certificate trust list (CTL)

Une liste prédéfinie d'éléments qui ont été signés par une entité de confiance. Un CTL peut contenir n'importe quels éléments, par exemple une liste de hachage de certificats, ou une liste de noms de fichiers. Tous les éléments d'une telle liste sont authentifiés (approuvés) par l'entité de signature.

Certification authority (CA)

Une autorité de certification est une organisation de confiance qui délivre les certificats affirmant qu'une entité, un individu, un ordinateur ou une organisation demandant un certificat remplit bien les conditions. C'est à dire qu'elle vérifie que l'entité est bien celle qu'elle prétend être.

Public Key Cryptography Standards (PKCS)

Ensemble de spécifications de la cryptographie à clé publique englobant les fonctions de sécurité, y compris les méthodes de signature de données, l'échange de clés, la demande de certificats, le chiffage et le déchiffage de clé publique ainsi que d'autres fonctions de sécurité.

On utilise par exemple PKCS#10 pour demander un nouveau certificat et PKCS#7 pour demander son renouvellement.


Voir aussi

Infrastructure à clé publique (PKI)

De manière générale, ensemble des réglementations, normes et logiciels permettant de réguler ou manipuler les certificats et les clés publiques ou privées.

Glossaire Microsoft

12 - Liens concernant la signature de code

Vous trouverez de nombreuses informations détaillées sur le sujet dans le mémoire suivant de Charles CHEBLI intitulé " **Signature et Chiffrement**".

Tutoriel Developpez.com :

Cryptographie : Signer des données avec l'API Windows

Divers :

 **How does the RemoteSigned execution policy work ?**

 **Microsoft Certificate Server and IIS 5.**

 **Internet Information Service (IIS) Server Certificate Installation Instructions.**

 **Microsoft Windows XP: Using Software Restriction Policies to Protect Against Unauthorized Software**

 **Configuring and Troubleshooting Certificate Services Client-Credential Roaming**

 **Signature de fichier .exe**


 **Signature d'un script WSH**

 **Signature de macros VBA**

 **Sauvegarde et restauration des certificats EFS**

 **Droits français**

 **FAQ sur les décrets** relatifs à la signature électronique.

 **Autoformation** sur la signature numérique.

