S3 - Atelier Gestion des utilisateurs - PowerShell

1. Gestion de la sécurité de lancement PowerShell

- Dans PowerShell, il existe 4 paramètres de stratégie d'exécution des scripts qui sont :
  - Restricted : paramètre par défaut, n'autorise pas l'exécution des scripts,
  - AllSigned : n'exécute que les scripts de confiance, donc signés,
  - RemoteSigned : exécute les scripts locaux sans obligation de confiance et les scripts de confiance issus d'Internet,
  - Unrestricted : autorise l'exécution de tous les scripts.
- Démarrer une console PowerShell en tant qu'administrateur :
  - Commande pour connaître la stratégie en cours :

    ```
    Get-ExecutionPolicy
    ```

  - Commande pour modifier la stratégie :

    ```
    Set-ExecutionPolicy -Scope LocalMachine -ExecutionPolicy Unrestricted
    ```

2. Accès aux comptes locaux du système

- Afficher les comptes locaux:

  ```
  Get-LocalUsers
  ```

- Ecrire le script suivant:

  ```
  $Nom = Read-Host -Prompt "Saisir un nom de compte local"
  $Compte = [ADSI]"WinNT://./$Nom"
  If ($Compte.Path)
  {
    Write-Host $Compte.FullName
  }
  Else
  {
    Write-Host "$Nom non trouvé"
  }
  ```

  > *Remarque: L'accès à la base locale de comptes utilisateurs d'un système Windows est réalisé avec l'instruction : [ADSI]"WinNT://." Ici, les majuscules et les minuscules doivent être respectées. Le point représente le nom du système sur lequel est lancée l'instruction, il peut être remplacé par le nom de l'ordinateur cible. Il faut bien sûr avoir des droits d'administration sur l'ordinateur distant. Pour filtrer les éléments de la base de comptes, il est possible de spécifier le nom de l'élément recherché, ici il est contenu dans la variable $nom : [ADSI]"WinNT://./$nom"*

- Tester le script:
  - Avec un compte inexistant
  - Avec un compte existant (vu avec l'affichage des comptes locaux)
- Une fois le test fait avec un compte existant, afficher les propriétés de la variable $compte

```
$compte | Get-Member
```

- Dans le volet de sortie, on peut voir les methodes et propriétés de la variable $Compte.
- Modifier le script pour afficher 3 autres propriétés.

3. Ajout d'un compte local

- Taper le script `AjoutCompteLocal.ps1` suivant:

```
##Ajoute un compte dans la base local du système
$Local = [ADSI]"WinNT://."
$Nom = Read-Host -Prompt "Saisir un nom de compte local"
$Description = Read-Host -Prompt "Saisir une description"
$Compte = [ADSI]"WinNT://./$Nom"
If (!$Compte.Path)
{
  $Utilisateur = $Local.Create("User",$Nom)
  $Utilisateur.invokeSet("Description",$Description)
  $Utilisateur.commitChanges()
  Write-Host "$Nom ajouté"
}
Else
{
  Write-Host "$Nom existe déjà"
}
```

> *Remarques: La variable `$local` possède une :**méthode** `create()` qui permet de créer un **objet** de type "user" (premier paramètre), dont le nom du compte est spécifié par le deuxième paramètre, ici la variable `$nom`. Le résultat est une variable nommée `$utilisateur`. La variable `$utilisateur` possède une **méthode** `InvokeSet()` qui permet de renseigner une information du compte, spécifiée par le premier paramètre, dont la valeur est contenue dans le deuxième, ici `$description`.*

- Tester le script en ajoutant un compte avec une description associée
  - Exemple: compte: Wilder1 description: test de création de compte
- Afficher les comptes locaux pour vérifier que le compte est bien crée
- Modifier le script pour ajouter d'autres propriétés
- Relancer le script pour vérifier que le compte est bien créé

4. Ajout d'utilisateurs par fichier

- Ecrire le script suivant:

```
# Création de dossier temporaire
$Dossier = "DossierTemporaire"
If (Test-Path "c:\$Dossier")
{
  Write-Host "Le dossier $Dossier existe déjà"
}
Else
{
  New-Item -Path "c:\$Dossier" -ItemType Directory
}
```

> *Ce script va créer automatiquement un dossier sous la racine c: si ce dossie n'existe pas. On utilise la variable `$Dossier` qui contient le nom du dossier.*

- Lancer le script et créer un dossier
- Télécharger le fichier comptes1.txtr et le copier dans le dossier que vous venez de créer.

> *Les informations du fichier **`comptes1.txt`** sont sous la forme : nomCompte/nomComplet/Description*

- Ecrire le script `LectureFichier.ps1` suivant:

```
# Parcours d'un fichier texte
#Emplacement du fichier texte
$Fichier = "c:\DossierTemporaire\comptes1.txt" #Mettre ici l'emplacement du fichier sous la forme "c:/..."
If (Test-Path $Fichier)
{
  $Lignes = Get-Content -Path $Fichier
  Foreach ($Ligne in $Lignes)
  {
      $TabCompte = $Ligne.Split("/")
      Write-Host $TabCompte[0]
  }
}
Else
{
  Write-Host "Fichier $Fichier non-trouvé"
}
```

> *Remarques : Cmdlet `Foreach` : Cette instruction peut s'interpréter de la manière suivante: Pour chaque ligne `$Ligne` contenue dans l'ensemble des lignes `$Lignes` la variable `$ligne` va successivement prendre la valeur de chaque ligne du tableau `Lignes`. `$ligne` est une chaîne de caractères. La variable `$ligne` possède une méthode `Split()` qui permet de retourner un tableau construit à partir de la chaîne de caractères contenue dans `$ligne`.*

> *Les éléments du tableau correspondent aux chaînes de caractères délimitées par le séparateur `"/"`.*

- Lancer le script
- Par rapport à ce qui vient de s'afficher, modifier le script pour que le nom complet et la description soient également affichés en dessous du nom du compte.
- Relancer le script
- En utilisant les 2 scripts `AjoutCompteLocal.ps1` et `LectureFichier.ps1` écrire un script qui permet d'ajouter dans la base locale du système, tous les comptes contenus dans le fichier Comptes1.txt.
- Tester ce nouveau script pour ajouter tous les comptes du fichier.
- Faire un deuxième test avec le même fichier, que se passe-t-il pour les noms déjà existants ?

5. Suppressions de comptes utilisateurs

- Ecrire le script suivant:

```
# Suppression de comptes dans la base locale
$Local = [ADSI]"WinNT://."
$Fichier = "c:\dossiertemporaire\comptes1.txt" #Mettre ici l'emplacement du
fichier sous la forme "c:/..."
If (Test-Path -Path $Fichier)
{
  $Lignes = Get-Content -Path $Fichier
  Foreach ($Ligne in $Lignes)
  {
      $TabCompte = $Ligne.Split("/")
      $Nom = $TabCompte[0]
      $Compte = [ADSI]"WinNT://./$Nom"
      If ($Compte.path)
      {
          $Local.delete("user",$Nom)
          Write-Host "$Nom supprimé"
      }
      Else
      {
          Write-Host "$Nom n'existe pas"
      }
  }
}
Else
{
  Write-Host "$Fichier non-trouvé"
}
```

> *Ce script permet de supprimer tous les comptes utilisateurs dont les noms sont contenus dans un fichier texte. Les informations sont toujours sous la forme :*

> *nomCompte/nomComplet/Description*

> *Remarques : Comme pour la création, la variable `$local` possède une méthode `delete()` qui permet de supprimer un objet de type "user" (premier paramètre), dont le nom du compte est spécifié par le deuxième paramètre, ici la variable `$nom`.*

- Executer le script et vérifier la liste des noms affichés
- Afficher les comptes pour vérifier que les comptes du fichier **Comptes1.txt** ont bien été supprimé.
- Modifier le fichier **Comptes1.txt** en y ajoutant les comptes crées au 3.
- Exécuter le script et vérifier si tous les comptes créés ont été supprimés.