Mr ROTH

Compte rendu TP7

20/03/2024

PowerShell – Import d’utilisateurs en masse

TEWES Arnaud
BTS SIO SISR 1ÈRE ANNÉE

1. Introduction

PowerShell est un langage de script qui s'utilise sur une interface en ligne de commande. En tant que langage de script, il est souvent utilisé pour automatiser certaines tâches de gestion des systèmes. Il sert également à créer, tester et à déployer des scripts d'automatisation de certaines tâches redondantes.
PowerShell a pour vocation de remplacer command.exe (ou cmd) et le langage Batch. Il est fondé sur de la programmation orientée objet. Le type de fichier PowerShell est le .ps1.

Sa syntaxe est composée de mots clés. Les commandes commencent toujours par un préfixe appelé « verbe ». Voici quelques exemples :

- Add – Pour ajouter des informations
- Get – Pour obtenir des informations (exemple : Get-Help nom de commande pour obtenir de l'aide sur une commande)
- New – Pour créer un nouvel objet ou une variable
- Set – Pour définir un nouvel objet ou une variable

Un script sert à indiquer à PowerShell où il doit aller chercher les informations, ce qu'il doit faire, et surtout comment le faire.

Prérequis :

- **Système d'exploitation** : PowerShell s'exécute sur Windows, Linux et macOS. (Natif sur Windows)
- Certains scripts nécessitent des modules supplémentaires qui doivent être installés en amont
- Certains scripts nécessitent une session PowerShell administrateur
- Avoir quelques connaissances de base de PowerShell

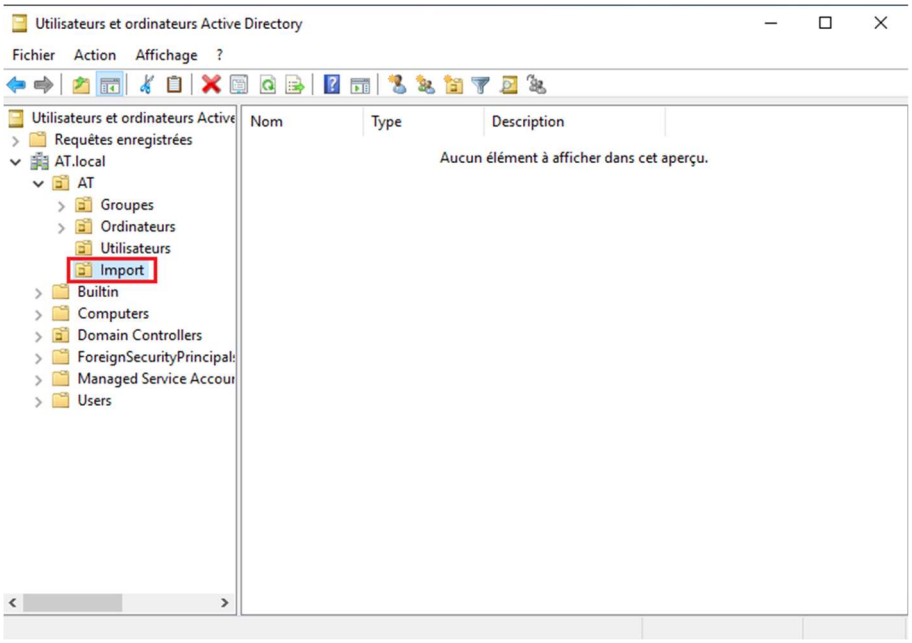
C'est un langage assez simple à comprendre et qui permet de faire absolument tout, et même plus de choses qu'avec une interface graphique. Grâce à lui, nous avons d'immenses possibilités, de l'automatisation à la création, jusqu'à déployer d'innombrables objets en un seul clic (une fois que le script est créé et fonctionnel)

Dans ce TP, je vais utiliser la force de PowerShell pour faire un déploiement de nouveaux utilisateurs en masse via un fichier .csv sur Active Directory.

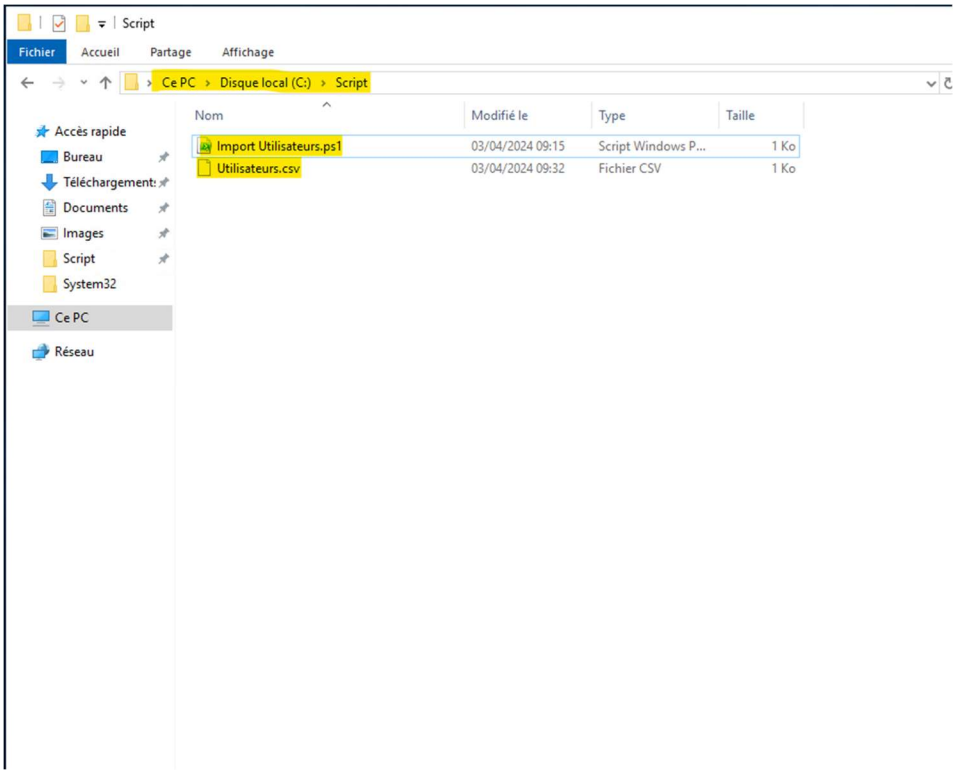
Un fichier .CSV (Comma-Separated Values) est un type de fichier qui est utilisé pour stocker des données sous forme de tableau (c'est-à-dire des données organisées en colonnes et en lignes). Chaque ligne du fichier correspond à une ligne de données, et chaque champ (ou colonne) est séparé par une virgule (ou un point-virgule dans certains cas). Si vous ouvrez un fichier .csv dans un tableur, il vous remontera les colonnes et les lignes telles qu'elles sont dans le fichier.

2. Procédé pas à pas de la création du .CSV et de la mise en place du script

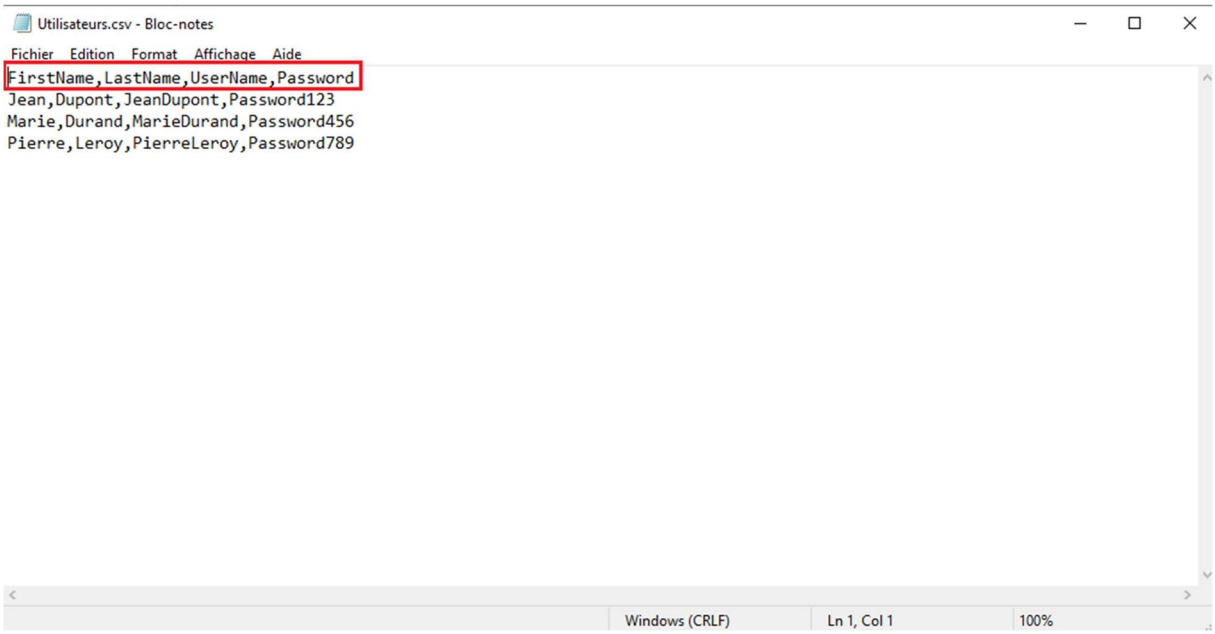
Dans un premier temps, j'ai créé une unité d'organisation dédiée « Import ». C'est dans ce dernier que je veux que mes utilisateurs créer via le script se range pour être sûr de ne pas désorganiser les autres UO si le script ne fonctionne pas (ce que je vous conseille de faire)



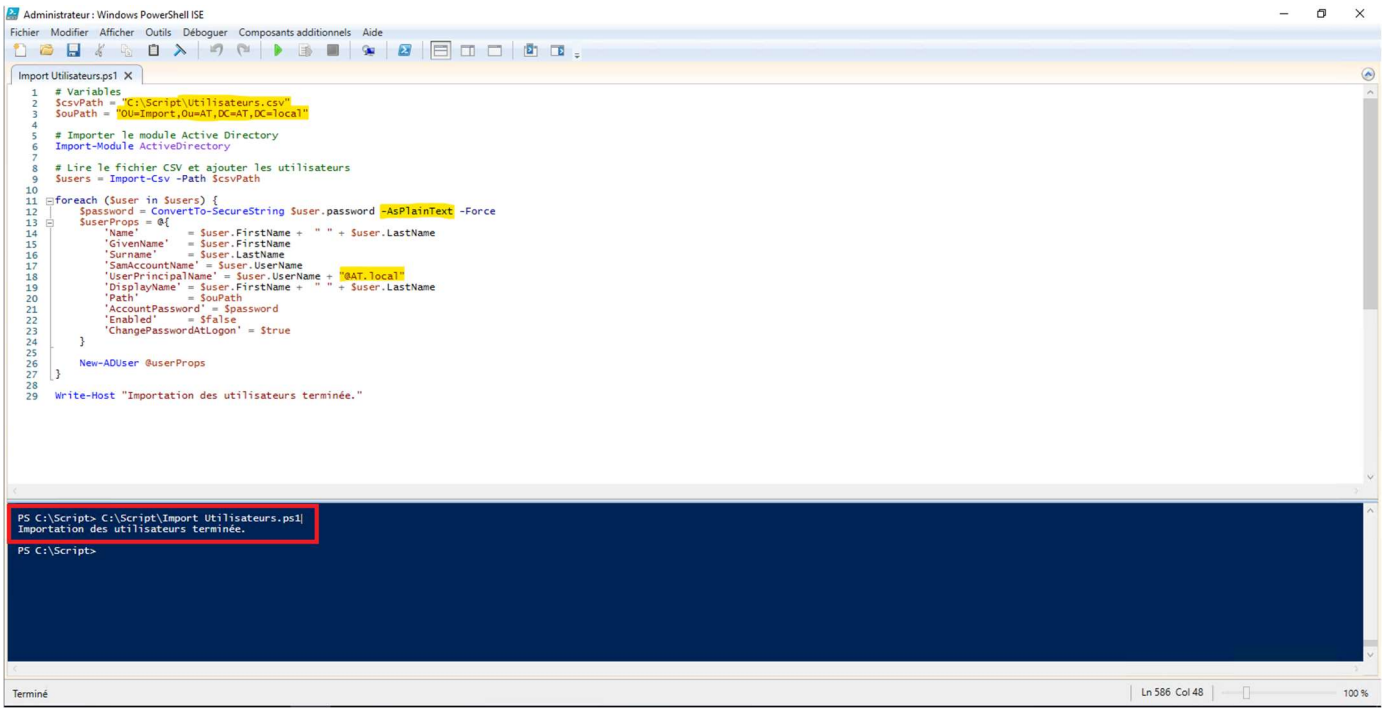
Ensuite, j'ai créé un dossier « Script » à la racine de mon disque local C : dans lequel je vais ranger mes scripts. Dans ce dernier, je crée mon fichier .csv (un bloc note par exemple que je renomme bien .csv à la fin pour qu'ils prennent cette extension)



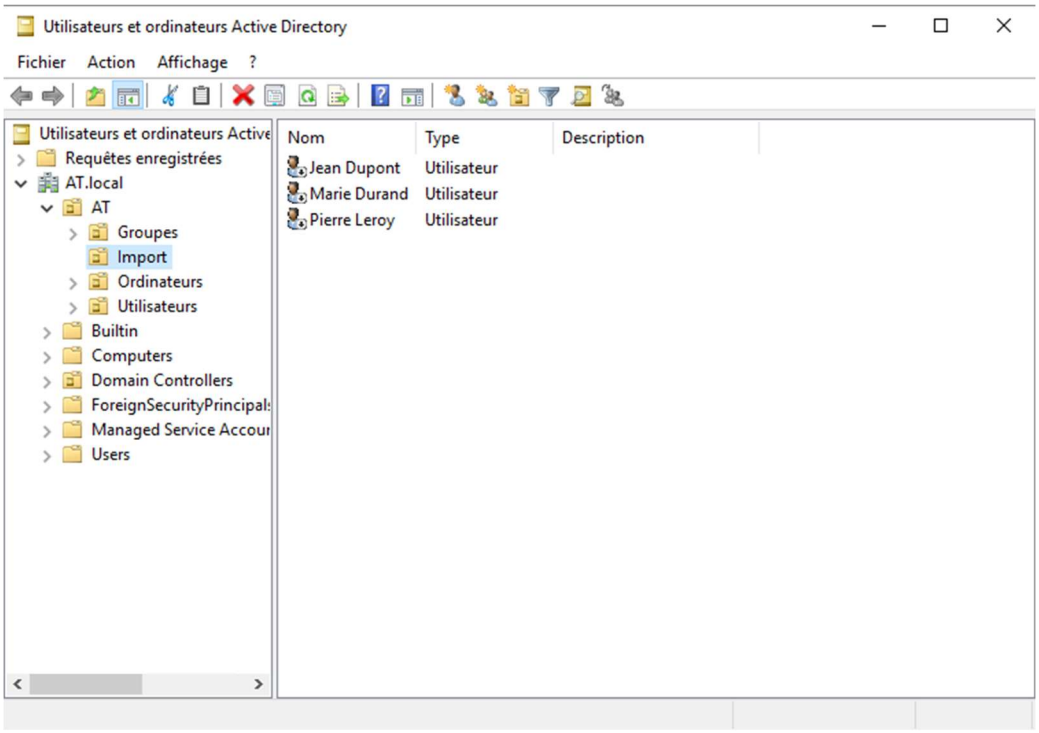
Voici par exemple, le fichier .csv que j'ai créé dans mon cas. La première ligne étant la première ligne de notre tableau, et chaque colonne étant séparée par une virgule. (Je n'ai créé que 3 utilisateurs pour faire ce test, mais j'aurais pu en mettre énormément plus)



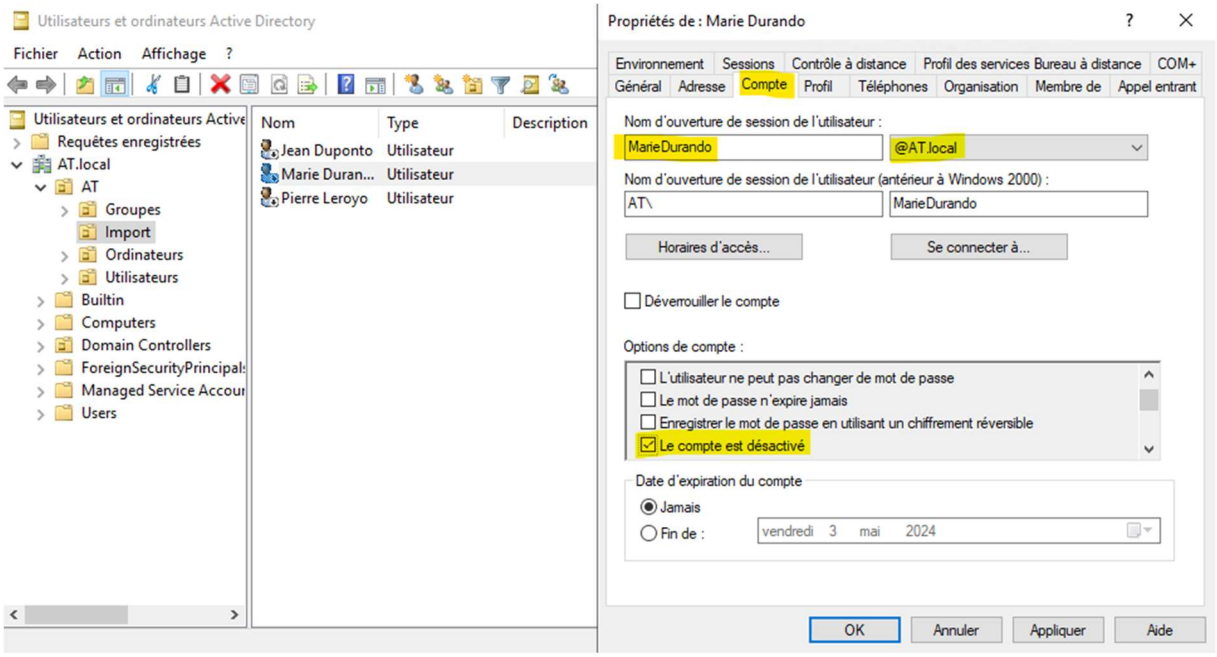
Ensuite, il nous faudra écrire un script en indiquant à PowerShell ou il doit aller chercher les informations, ce qu'il doit faire, et surtout comment le faire. C'est à cela que sert un script. Il faudra bien évidemment des connaissances pour comprendre le script. Dans notre cas, nous avons utilisé un script directement généré via l'IA. Je détaillerai plus ce que fait le script dans la conclusion et le mettrai à disposition. Je vous invite à aller voir cette dernière si vous vous posez des questions sur son fonctionnement



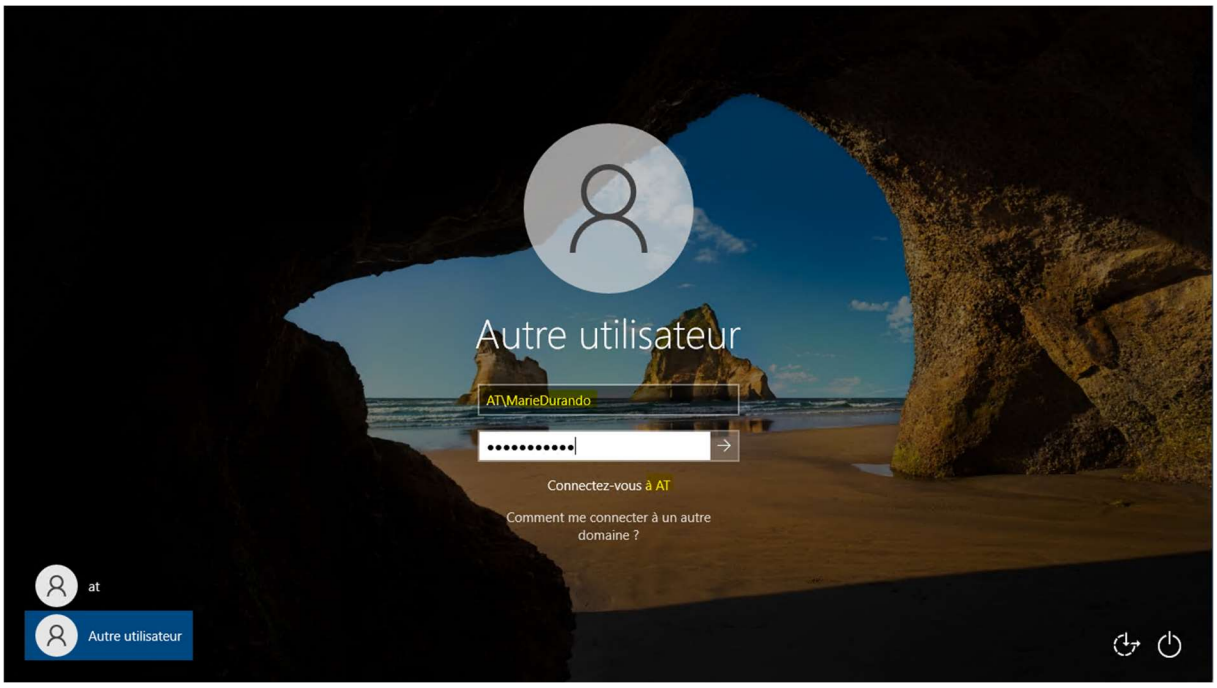
Nous voyons que la console de PowerShell nous indique que le script est terminé et que les utilisateurs ont correctement été importés. S'il y avait eu une erreur dans le script, c'est dans cette même console que nous aurions eu les informations d'erreur (ou log).
Nous allons maintenant vérifier que les utilisateurs ont bien été créés dans notre UO Import créé précédemment.



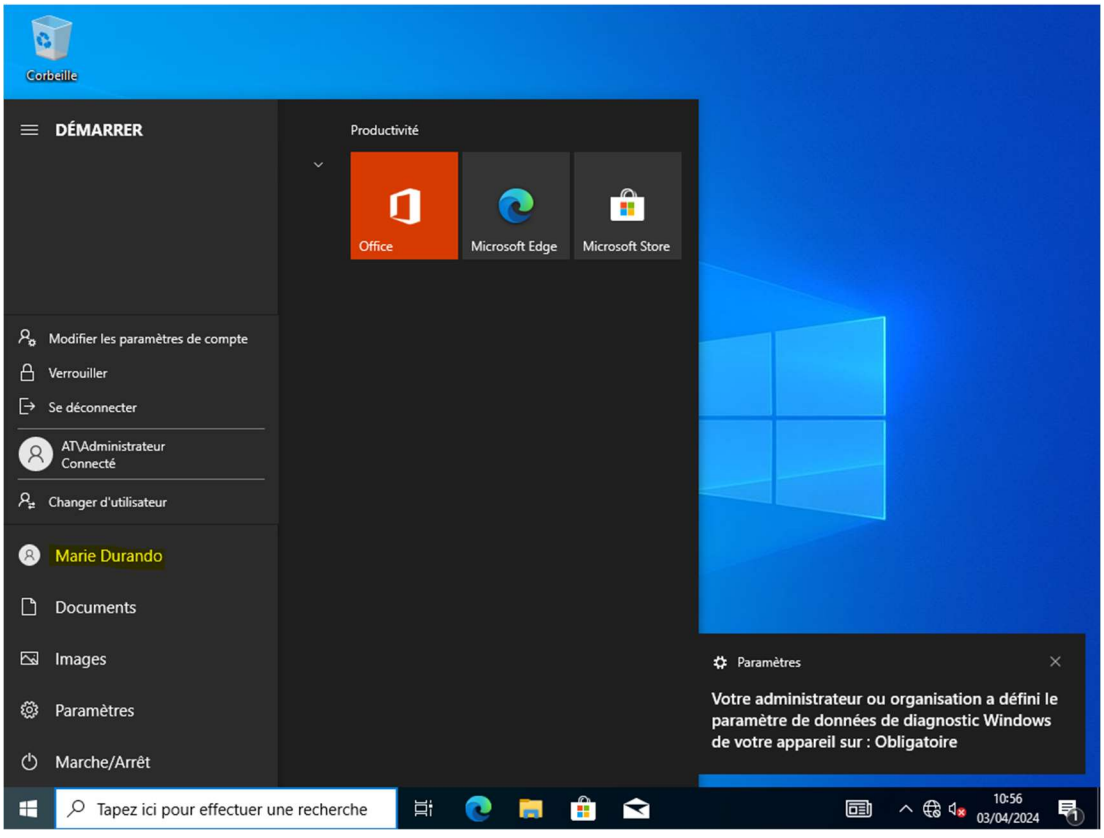
Et comme je l'avais programmé dans le script, les utilisateurs sont bien désactivés à la création (pour une sécurité plus optimale)



Je vais maintenant tester un des comptes créer pour vérifier que ces derniers fonctionnent correctement (en prenant soin d'activer le compte avant)



Nous sommes bien connectés sur la session créée précédemment via le script PowerShell et le fichier .csv



3. Conclusion

Et voilà, nous avons ajouté des utilisateurs via un script PowerShell et un fichier .csv.

C'est une des innombrable fonctionnalités de PowerShell. Ce dernier est un outil ultra puissant et moderne capable d'automatiser des tâches, ou comme dans le cas présent, créer des utilisateurs en masse via un simple fichier .csv.

Test de bon fonctionnement :

Il suffit de lancer le script et de vérifier dans nos utilisateurs Active Directory si les utilisateurs ont bien été créés. S'il y avait une erreur, nous aurions eu l'information dans la console de PowerShell lorsque le script est terminé. Nous pouvons également se connecter sur un des comptes créés via le script pour vérifier que le compte fonctionne correctement

Points de vigilances et conseils de sécurité :

- Dans un réel d'import d'utilisateurs en masse, bien laisser l'utilisateur crée via le script désactivé et lui demander de changer son mot de passe à la première connexion, ce qui évite le risque d'usurpation (ou qu'il devienne moindre)

- Le fichier CSV avec les mots de passe des utilisateurs en clair étant un fichier assez sensible, ne pas oublier de bien le supprimer pour éviter les soucis.

- Bien utiliser une UO dédiée pour la création des utilisateurs avec un script. Dans le cas d'une erreur, c'est plus facilement nettoyable et permet de ne pas désorganiser les UOs déjà existantes

- Si le script ne fonctionne pas, c'est qu'il à certainement une erreur sur une des lignes. La console PowerShell indique normalement d'où vient l'erreur et peut vous permettre de la corriger au besoin

Voici également le script PowerShell utilisé, à adapter pour votre cas précis :

```
# Variables

$csvPath = "C:\Script\Utilisateurs.csv" # Définit le chemin vers le fichier CSV contenant les informations des utilisateurs.

$ouPath = "OU=Import,Ou=AT,DC=AT,DC=local" # Définit le chemin de l'unité d'organisation (OU) où les nouveaux utilisateurs seront créés.


# Importer le module Active Directory

Import-Module ActiveDirectory # Importe le module Active Directory


# Lire le fichier CSV et ajouter les utilisateurs

$users = Import-Csv -Path $csvPath  # Lit le fichier CSV et stocke les informations des utilisateurs dans la variable $users.


foreach ($user in $users) { # Pour chaque utilisateur dans le fichier CSV...

   $password = ConvertTo-SecureString $user.password -AsPlainText -Force # Convertit le mot de passe en texte brut en une chaîne sécurisée.

   $userProps = @{ # Crée un tableau associatif pour stocker les propriétés de l'utilisateur.

      'Name'        = $user.FirstName +  " " + $user.LastName # Le nom complet de l'utilisateur.

      'GivenName'   = $user.FirstName # Le prénom de l'utilisateur.

      'Surname'     = $user.LastName # Le nom de famille de l'utilisateur.

      'SamAccountName' = $user.UserName # Le nom d'utilisateur SAM (Security Accounts Manager).

      'UserPrincipalName' = $user.UserName + "@AT.local" # Le nom principal de l'utilisateur (UPN).

      'DisplayName' = $user.FirstName +  " " + $user.LastName # Le nom d'affichage de l'utilisateur.

      'Path'        = $ouPath # Le chemin de l'OU où l'utilisateur sera créé.

      'AccountPassword' = $password # Le mot de passe de l'utilisateur.

      'Enabled'     = $false # Le compte de l'utilisateur est désactivé par défaut.

      'ChangePasswordAtLogon' = $true # L'utilisateur doit changer son mot de passe lors de sa première connexion.

   }

   New-ADUser @userProps # Crée un nouvel utilisateur dans Active Directory avec les propriétés définies ci-dessus.

}

Write-Host "Importation des utilisateurs terminée." # Affiche un message indiquant que l'importation des utilisateurs est terminée.
```