

Développement d'un Script PowerShell pour la Création Automatisée de Profils Utilisateur dans Active Directory

Introduction : Au cours de mon stage, j'ai été chargé de développer un script PowerShell visant à automatiser la création de plus d'une centaine de profils utilisateur pour les élèves dans le domaine Active Directory. Ce projet visait à simplifier et accélérer le processus de gestion des comptes utilisateurs en utilisant des données fournies dans un fichier Excel.

Étapes du Projet :

Analyse des Besoins :

Compréhension des exigences de l'entreprise concernant la création de profils utilisateur.

Identification des informations nécessaires pour chaque profil utilisateur : prénom, nom, mot de passe.

Détermination du format du nom d'utilisateur : première lettre du prénom suivie d'un point et du nom de famille.

Préparation des Données :

Collecte des données nécessaires sous forme de fichier Excel contenant les colonnes : prénom, nom, mot de passe.

Vérification et nettoyage des données pour éviter les duplicatas et les erreurs.

Conception du Script PowerShell :

Écriture du script PowerShell pour automatiser la lecture du fichier Excel.

Utilisation de la bibliothèque PowerShell ImportExcel pour manipuler les données Excel.

Génération des noms d'utilisateur selon le format défini (initiale du prénom suivie d'un point et du nom de famille).

Développement et Test du Script :

Importation des modules nécessaires et lecture du fichier Excel :

Boucle à travers les données pour créer les comptes utilisateur dans Active Directory :

Test du script sur un environnement de développement pour vérifier son bon fonctionnement.

Mise en Production :

Exécution du script dans l'environnement de production.

Vérification de la création correcte des comptes et résolution des éventuels problèmes.

Documentation et Formation :

Rédaction d'une documentation détaillée sur le script, son utilisation et sa maintenance.

Formation de l'équipe IT sur l'utilisation et les modifications possibles du script.

Résultats et Bénéfices : Le script PowerShell développé a permis de créer rapidement et efficacement les comptes utilisateurs pour les élèves, réduisant considérablement le temps et les efforts nécessaires par rapport à une création manuelle. La standardisation des noms d'utilisateur a également facilité la gestion et la maintenance des comptes dans Active Directory.

Conclusion : Ce projet m'a permis de renforcer mes compétences en scripting PowerShell et en administration de systèmes. J'ai également appris à gérer des données à partir de fichiers Excel et à automatiser des processus répétitifs, contribuant ainsi à améliorer l'efficacité des opérations IT.

Remerciements : Je tiens à remercier toute l'équipe pour leur soutien et leurs conseils tout au long de ce projet. Leur aide a été précieuse pour mener à bien cette mission.

VOIR OU SE TOUVE UN USER.ps1

Création-utilisateur-OU-élève.ps1

nombre utilisateur dans l'OU.ps1 X

```
1 # Spécifiez le nom du domaine
2 $domain = "hanned.lan"
3
4 # Spécifiez le chemin complet de l'OU "élève"
5 $ouPath = "OU=élève,DC=hanned,DC=lan"
6
7 # Obtenez le nombre d'utilisateurs dans l'OU "élève"
8 $userCount = (Get-ADUser -Filter * -SearchBase $ouPath).Count
9
10 # Affichez le nombre d'utilisateurs dans l'OU "élève"
11 Write-Host "Le nombre d'utilisateurs dans l'OU 'élève' du domaine '$domain' est : $userCount"
12
```

Le prénom ou le nom est manquant pour un utilisateur.
Le prénom ou le nom est manquant pour un utilisateur.
Le prénom ou le nom est manquant pour un utilisateur.
Le prénom ou le nom est manquant pour un utilisateur.
Le prénom ou le nom est manquant pour un utilisateur.
Le prénom ou le nom est manquant pour un utilisateur.
Le prénom ou le nom est manquant pour un utilisateur.
Le prénom ou le nom est manquant pour un utilisateur.

PS C:\Windows\system32> C:\Users\Administrateur\Downloads\nombre utilisateur dans l'OU.ps1
Le nombre d'utilisateurs dans l'OU 'élève' du domaine 'hanned.lan' est : 152

PS C:\Windows\system32>

[illegible]



Utilisateurs et ordinateurs Active Directory

- > Requêtes enregistrées
- ▼ hanned.lan
 - > Built-in
 - > Computers
 - > Domain Controllers
 - élève
 - Enseignants
 - > ForeignSecurityPrincipals
 - > Managed Service Accounts
 - > ordinateur
 - > secretariat
 - > Users
 - > utilisateur sans mot de passe

Nom	Type	Description
Abdallah DJ...	Utilisateur	
Abdarrahma...	Utilisateur	
Abdelchahid...	Utilisateur	
Abdelkarim ...	Utilisateur	
Abdelrahma...	Utilisateur	
Abderahma...	Utilisateur	
Abderahma...	Utilisateur	
Abderahmè...	Utilisateur	
Abd-Errahm...	Utilisateur	
Abdoulah J...	Utilisateur	
Adam BOUC...	Utilisateur	
Ahmed MA...	Utilisateur	
Ahmed PAO...	Utilisateur	
Abdelkarim ...	Utilisateur	

VOIR OU SE TROUVE UN USER.ps1

Création-utilisateur-OU-élève.ps1

nombre utilisateur dans l'OU.ps1

```
1 # Spécifiez le chemin d'accès du fichier Excel contenant les informations des utilisateurs
2 $excelFilePath = "E:\telechargements\test2.xlsx"
3
4 # Vérifiez si le module ImportExcel est installé, sinon, installez-le
5 if (-not (Get-Module -ListAvailable -Name ImportExcel)) {
6     Install-Module -Name ImportExcel -Force -Scope CurrentUser
7 }
8
9 # Chargez le module ImportExcel
10 Import-Module -Name ImportExcel
11
12 # Chargez les données du fichier Excel
13 $users = Import-Excel -Path $excelFilePath
14
15 # Parcourez chaque utilisateur dans le fichier Excel
16 foreach ($user in $users) {
17     # Vérifiez si le prénom et le nom ne sont pas nuls ou vides
18     if (-not [string]::IsNullOrEmpty($user.Prénom) -and -not [string]::IsNullOrEmpty($user.Nom)) {
19         # Récupérez le nom et le prénom de l'utilisateur
20         $nom = $user.Nom.Trim()
21         $prenom = $user.Prénom.Trim()
22
23         # Créez le nom d'utilisateur en utilisant la première lettre du prénom et le nom de famille
24         $username = ($prenom.Substring(0, 1) + "." + $nom).ToLower()
25
26         # Vérifiez si le mot de passe n'est pas nul ou vide
27         if (-not [string]::IsNullOrEmpty($user.MotDePasse)) {
28             $password = $user.MotDePasse
29
30             # Vérifiez si l'utilisateur n'existe pas déjà
31             $existingUser = Get-ADUser -Filter {SamAccountName -eq $username} -ErrorAction SilentlyContinue
32             if (-not $existingUser) {
33                 # Créez l'utilisateur avec le nom d'utilisateur et le mot de passe
34                 New-ADUser -Name "$prenom $nom" -SamAccountName $username -GivenName $prenom -Surname $nom -AccountPassword (ConvertTo-SecureString -AsPlainText $password -Force) -Path "OU=
35             } else {
36                 Write-Host "Utilisateur $username créé avec succès."
37             }
38         } else {
39             Write-Host "Le mot de passe est manquant pour l'utilisateur $username."
40         }
41     } else {
42         Write-Host "Le prénom ou le nom est manquant pour un utilisateur."
43     }
44 }
45 }
```

```
L'utilisateur n.tiroutine existe déjà.
L'utilisateur m.vignal existe déjà.
L'utilisateur a.abdelmaaboud existe déjà.
L'utilisateur t.arar existe déjà.
L'utilisateur s.baouahi existe déjà.
L'utilisateur i.belamri existe déjà.
L'utilisateur n.boucharafat existe déjà.
L'utilisateur r.boufenzli existe déjà.
L'utilisateur a.diakite existe déjà.
L'utilisateur m.diawara existe déjà.
L'utilisateur c.el iraki existe déjà.
L'utilisateur r.ferrag existe déjà.
L'utilisateur h.frnina existe déjà.
```

Commandes

Modules

Tout

Nom :

A:

- Add-ADCentralAccessPolicyMember
- Add-ADComputerServiceAccount
- Add-ADDomainControllerPasswordReplication
- Add-ADDsReadOnlyDomainControllerAccount
- Add-ADFineGrainedPasswordPolicySubject
- Add-ADGroupMember
- Add-ADPrincipalGroupMembership
- Add-ADResourcePropertyListMember
- Add-AppvClientConnectionGroup
- Add-AppvClientPackage
- Add-AppvPublishingServer
- Add-AppxPackage
- Add-AppxProvisionedPackage
- Add-AppxVolume
- Add-BCDataCacheExtension
- Add-BitsFile
- Add-CertificateEnrollmentPolicyServer
- Add-ClusterSCSITargetServerRole
- Add-Computer
- Add-ConditionalFormatting
- Add-Content
- Add-DfsrConnection
- Add-DfsrMember
- Add-DhcpServerInDC
- Add-DhcpServerSecurityGroup
- Add-DhcpServ4Class
- Add-DhcpServ4ExclusionRange
- Add-DhcpServ4Failover
- Add-DhcpServ4FailoverScope
- Add-DhcpServ4Filter
- Add-DhcpServ4Lease
- Add-DhcpServ4MulticastExclusionRange
- Add-DhcpServ4MulticastScope
- Add-DhcpServ4OptionDefinition
- Add-DhcpServ4Policy
- Add-DhcpServ4PolicyIPRange
- Add-DhcpServ4Reservation
- Add-DhcpServ4Scope
- Add-DhcpServ4Superscope
- Add-DhcpServ6Class
- Add-DhcpServ6ExclusionRange