

IUT DE COLMAR

R405 AUTOMATISATION DES TÂCHES

ANNÉE 2022-23

TP 2 : Objet Logs et utilisateurs

MARTIN BAUMGAERTNER

17 mars 2023

Table des matières

1	Gestion locale des utilisateurs	2
1.1	Exercice 1 - la liste des utilisateurs de la machine	2
1.2	Exercice 2 - Après avoir créer un nouvel utilisateur avec Powershell, dans quel groupe celui-ci est présent ?	2
1.3	Exercice 3	2
1.4	Exercice 4 - Écrire un script qui affiche le nom d'utilisateur avec la date de sa dernière connexion	4
1.5	Exercice 5	4
2	Gestion des logs	5
2.1	Exercice 6 - Quelle option permet d'afficher la liste des logs dispo- nibles ?	5
2.2	Exercice 7 - Afficher les sources du log système, sans doublon de préférence	5
2.3	Exercice 8	5

1 Gestion locale des utilisateurs

1.1 Exercice 1 - la liste des utilisateurs de la machine

Pour dresser la liste des utilisateurs il faut faire la commande **Get-LocalUser**. J'en ai donc au total 5, qui sont **Administrator**, **DefaultAccount**, **Invité**, **toto** et **WDAGUtilityAccount**.

1.2 Exercice 2 - Après avoir créer un nouvel utilisateur avec Powershell, dans quel groupe celui-ci est présent ?

L'utilisateur a été créé avec la commande **New-LocalUser**, suite à ça un nom d'utilisateur est demandé, puis un mot de passe.

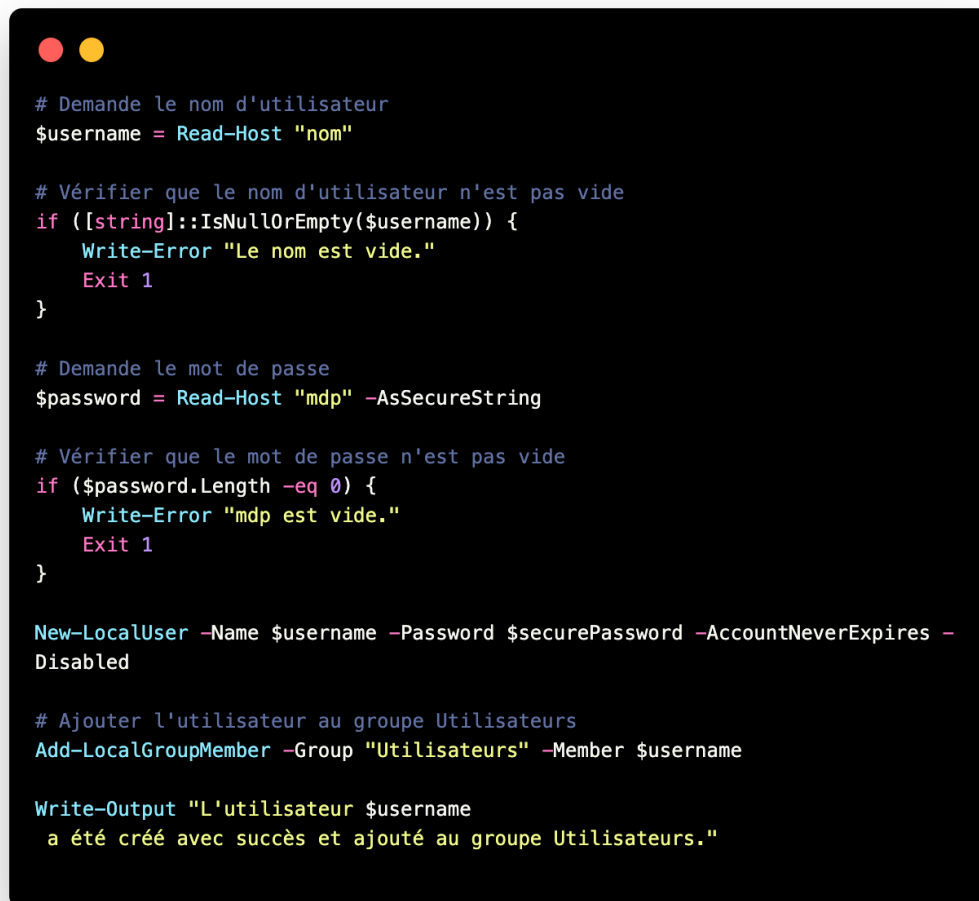
L'utilisateur créé n'a pas de groupe affecté, car nous n'avons pas, utilisé la commande suivante **Add-LocalGroupMember**, suite à sa création.

1.3 Exercice 3

Consigne

Créer un script qui crée un utilisateur dans le groupe Utilisateurs, le nom et le mot de passe sont demandés par le script et le compte sera désactivé par défaut. L'absence de nom d'utilisateur et de mot de passe seront gérés par l'affichage d'un message d'erreur. Pour l'entrée du mot de passe l'option **-AsSecureString** peut être mise à la commande **Read-Host**.

Voici ci-dessous le script demandé. Le paramètre **-AsSecureString** permet de transformer le champs basique string en un champs de type SecureString, ce qui ne donnera donc pas d'erreur et l'utilisateur pourra être créé.



```
# Demande le nom d'utilisateur
$username = Read-Host "nom"

# Vérifier que le nom d'utilisateur n'est pas vide
if ([string]::IsNullOrEmpty($username)) {
    Write-Error "Le nom est vide."
    Exit 1
}

# Demande le mot de passe
$password = Read-Host "mdp" -AsSecureString

# Vérifier que le mot de passe n'est pas vide
if ($password.Length -eq 0) {
    Write-Error "mdp est vide."
    Exit 1
}

New-LocalUser -Name $username -Password $securePassword -AccountNeverExpires -
Disabled

# Ajouter l'utilisateur au groupe Utilisateurs
Add-LocalGroupMember -Group "Utilisateurs" -Member $username

Write-Output "L'utilisateur $username
a été créé avec succès et ajouté au groupe Utilisateurs."
```

FIGURE 1 – Script création d'utilisateur

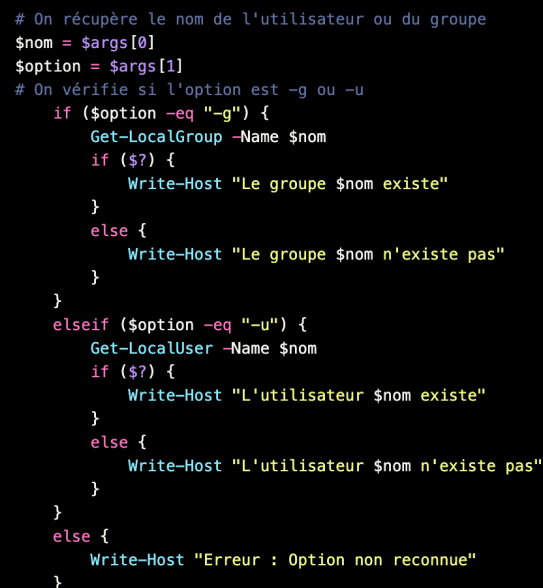
1.4 Exercice 4 - Écrire un script qui affiche le nom d'utilisateur avec la date de sa dernière connexion

Pour pouvoir afficher la dernière connexion des utilisateurs il suffit de faire la commande suivante **Get-LocalUser | Select-Object Name, LastLogOn**. Le script nous renverra alors un tableau avec les noms d'utilisateurs et leurs date et heure de dernière connexion.

1.5 Exercice 5

Consigne

Écrire un script qui se nomme `verifie_doublon.ps1` et qui vérifie si le groupe ou l'utilisateur existe. Le programme prendra en paramètre le nom d'utilisateur ou de groupe que l'on pourra distinguer avec une option (-g pour groupe ou -u pour utilisateur). Voici donc ci-après le script demandé, avec le bon choix de l'option en fonction -g ou -u...



```
# On récupère le nom de l'utilisateur ou du groupe
$nom = $args[0]
$option = $args[1]
# On vérifie si l'option est -g ou -u
if ($option -eq "-g") {
    Get-LocalGroup -Name $nom
    if ($?) {
        Write-Host "Le groupe $nom existe"
    }
    else {
        Write-Host "Le groupe $nom n'existe pas"
    }
}
elseif ($option -eq "-u") {
    Get-LocalUser -Name $nom
    if ($?) {
        Write-Host "L'utilisateur $nom existe"
    }
    else {
        Write-Host "L'utilisateur $nom n'existe pas"
    }
}
else {
    Write-Host "Erreur : Option non reconnue"
}
```

FIGURE 2 – `verifie_doublon.ps1`

2 Gestion des logs

2.1 Exercice 6 - Quelle option permet d'afficher la liste des logs disponibles ?


La commande permettant d'afficher les logs disponibles est **Get-EventLog -List**

2.2 Exercice 7 - Afficher les sources du log système, sans doublon de préférence

Pour afficher les sources du log système comme demandé il suffit de faire la commande suivante : **Get-EventLog -LogName System | Select-Object -Property Source -Unique**. Le script nous renverra alors une liste avec les sources du log système. L'option **-Unique** permet de supprimer les doublons.

En rajoutant **| Measure-Object** à la fin de la commande on peut voir qu'il y a 24 logs différents.

2.3 Exercice 8

A screenshot of a PowerShell script in a terminal window. The script is designed to retrieve system event logs, filter for unique sources, and export the results to a CSV file. The code uses `Get-EventLog` to fetch logs for the 'System' log name from the 'user32' source. It then creates a custom object table with properties 'Raison' (Message) and 'Date' (TimeGenerated). Finally, it uses `Export-CSV` to save the data to 'C:\Users\Administrateur\Downloads\Exercice8.csv' without type information.

```
$events = Get-EventLog -LogName System -Source user32
# création du tableau
$tab = @()
foreach ($event in $events) {
    $reason = $event.Message
    $date = $event.TimeGenerated
    $tab += New-Object PSObject -Property @{
        "Raison" = $reason
        "Date" = $date
    }
}
# export du tableau dans un fichier csv
$tab | Export-CSV -Path "C:\Users\Administrateur\Downloads\Exercice8.csv" -
NoTypeInfo
```

FIGURE 3 – Script CSV