Powershell / CMD

- Mail List
- Etat des lieux d'un service Windows
- Export Liste fichier
- Extract AD pc non connecté
- Suppression de dossier à partir d'un export
- AD extract exclusion
- Suppression user CSV
- Extract nom de dossier
- Crypt password + line to add on script
- Verifier le issuer du certificat

Mail List

```
import csv
import smtplib
from email.mime.text import MIMEText

# Define the email parameters
sender_email = 'your_email@example.com'
sender_password = 'your_email_password'
subject = 'Update your password'
message_template = '''\
Dear {name},

Please update your password for our service. You can do this by logging into your account and changing your password.

Thank you,
The Service Team
'''

# Read the CSV file and send an email to each user
with open('user_data.csv') as file:
    reader = csv.reader(file)
    next(reader)  # Skip header row
    for row in reader:
        # Extract user data
        name, email = row
        # Create the email message
        message = message_template.format(name=name)
        email_message = MIMEText(message)
        email_message['Subject'] = subject
        email_message['From'] = sender_email
        email_message['To'] = email
        # Connect to the SMTP server and send the email
        with smtplib.SMTP('smtp.gmail.com', 587) as server:
```

```
        server.starttls()
        server.login(sender_email, sender_password)
        server.sendmail(sender_email, email, email_message.as_string())
    print(f'Sent email to {name} ({email})')
```

Etat des lieux d'un service Windows

> Listage de l'état d'un service sur l'ensemble des serveurs Windows d'un domaine
> Le compte de la machine ne doit pas être désactivé et il doit répondre au ping
> Dans ce cas, il s'agit du service 'spooler'

```
# Autor : JHAF

# Service à check
$Service = 'spooler'

# Horadatage pour CSV
$DateJour = Get-Date -Uformat %Y%m%d_%H%m

# Initialisation des variables en Array
$Report = @()
$Spoolers = @()

# Liste des serveurs Windows avec une compte actif
$Servers = Get-ADOrganizationalUnit -Filter * | ForEach-Object { Get-ADComputer -Filter 'operatingSystem -like
"*Windows Server*" -and userAccountControl -notlike "4098"' -SearchBase "$_" -SearchScope OneLevel } |
Select-Object DNSHostName, Name, SID
#$Servers | Out-GridView

# Test si les serveurs sont UP
foreach($Server in $Servers){

   $pingtest = Test-Connection -ComputerName $Server.DNSHostName -Quiet -Count 1 -ErrorAction
SilentlyContinue
   if($pingtest){
      #Write-Host $Server.DNSHostName " is reachable"
      $Spoolers += Get-ADComputer -Identity $Server.Name | Select-Object DNSHostName, Name, SID
```

```
    }
    else{
      #Write-Host $Server.DNSHostName " is not reachable"
    }

}
#$Spoolers | Out-GridView

# Récupération des infos du service Spooler pour chaque serveur UP
foreach ($Spooler in $Spoolers) {
   $Status = Get-Service -ComputerName $Spooler.Name -Name $Service | Select-Object MachineName, Name,
Status
   $Startup = Get-WmiObject -Computer $Spooler.Name -Class Win32_Service -Property StartMode -Filter
"Name='$Service'"
   $Report += [PSCustomObject]@{
      MachineName = $Status.MachineName
      Name        = $Status.Name
      Status      = $Status.Status
      StartMode   = $Startup.StartMode
   }
}
$Report | Out-GridView
#$Report | Export-Csv -Path C:\script\Audit\2022\1032\${DateJour}_Get-Spooler_Status.csv -NoTypeInformation
-Encoding UTF8 -Delimiter ';'
```

Arrêt puis désactivation du démarrage auto d'un service sur l'ensemble des machines Windows listé dans un CSV
WinRM doit fonctionner depuis la machine qui lance le script (notamment les flux RPC doivent être ouverts)
Dans ce cas, il s'agit du service 'spooler'

```
# Horadatage pour CSV
$DateJour = Get-Date -Uformat %Y%m%d_%H%m

# Service à stop et check
$Service = 'spooler'
```

```
# Initialisation des variables en Array
$Report = @()

# Fichier CSV à faire à la main pour import
$SpoolersStop = Import-Csv C:\script\Audit\2022\1032\Stop-Spoolers.csv -Delimiter ';'

# Arrêt du service et Désactivation du démarrage auto
foreach ($Spooler in $SpoolersStop) {
    Get-Service -ComputerName $Spooler.MachineName -Name "spooler" | Stop-Service
    Set-Service -Computer $Spooler.MachineName -Name "spooler" -StartupType "Disabled"
}

# Récupération des infos du service Spooler pour chaque serveur UP
foreach ($Spooler in $Spoolers) {
    $Status = Get-Service -ComputerName $Spooler.Name -Name $Service | Select-Object MachineName, Name,
Status
    $Startup = Get-WmiObject -Computer $Spooler.Name -Class Win32_Service -Property StartMode -Filter
"Name='$Service'"
    $Report += [PSCustomObject]@{
        MachineName = $Status.MachineName
        Name        = $Status.Name
        Status      = $Status.Status
        StartMode   = $Startup.StartMode
    }
}
#$Report | Out-GridView
$Report | Export-Csv -Path C:\script\Audit\2022\1032\${DateJour}_Stop-Spoolers_Check.csv -NoTypeInformation
-Encoding UTF8 -Delimiter ';'
```

Export Liste fichier

```
$filepath = "E:\liste_fichiers_exe.csv"
 Get-ChildItem -Path E:\ -Recurse -Filter *.exe | Select-Object FullName | Export-Csv -Path $filepath -
NoTypeInformation
```

Extract AD pc non connecté

```
$DaysInactive = 60
$InactiveDate = (Get-Date).AddDays(-$DaysInactive)
$ExportFile = "C:\temp\InactiveComputers.csv"



$Computers = Get-ADComputer -Filter * -Properties LastLogonDate, Description



$InactiveComputers = $Computers | Where-Object { $_.LastLogonDate -lt $InactiveDate }

$InactiveComputerDataWithUser = foreach ($computer in $InactiveComputers) {
    $computerName = $computer.Name
    $description = $computer.Description

    $lastLogon = $computer.LastLogonDate
    $lastLogonUser = Get-ADUser -Filter "SamAccountName -eq '$($computer.SamAccountName)'" -Property
SamAccountName |
        Select-Object -ExpandProperty SamAccountName

    [PSCustomObject]@{
        ComputerName = $computerName
        Description = $description
        LastLogonDate = $lastLogon
        User = $lastLogonUser
    }
}

$ExportDescription = "List of inactive computers that have not logged on in the last $DaysInactive days."
$InactiveComputerDataWithUser | Export-Csv -Path $ExportFile -NoTypeInformation -Append
Add-Content -Path $ExportFile -Value "`nDescription: $ExportDescription"
```

Suppression de dossier à partir d'un export

```
$csvFilePath = "le fichier csv"
$folderNameColumn = "le nom de la colonne"
$directoryPath = "le chemin\"

# Read the CSV file
$csvData = Import-Csv -Path $csvFilePath

# Delete folders
$csvData | ForEach-Object {
    $folderName = $_.$folderNameColumn
    $folderPath = Join-Path -Path $directoryPath -ChildPath $folderName

    if (Test-Path -Path $folderPath -PathType Container) {
        try {
            Remove-Item -Path $folderPath -Recurse -Force
            Write-Host "Deleted folder: $folderName"
        }
        catch {
            Write-Host "Failed to delete folder: $folderName - $_"
        }
    }
    else {
        Write-Host "Folder not found: $folderName"
    }
}
```

AD extract exclusion

```
# Importer le module Active Directory
Import-Module ActiveDirectory

# Spécifier le chemin de sortie pour le fichier CSV
$cheminSortie = "C:\temp\utilisateurs.csv"

# Définir les filtres pour exclure les OU spécifiées
$ouExclues =
        "OU=***,DC=domaine,DC=com",
        "OU=***,DC=domaine,DC=com",
        "OU=***,DC=domaine,DC=com"

# Récupérer tous les utilisateurs en excluant les OU spécifiées et inclure le nom canonique de l'OU
$utilisateurs = Get-ADUser -Filter * -Property EmailAddress |
            Where-Object { $ouExclues -notcontains $_.DistinguishedName } |
            Select-Object SamAccountName, GivenName, Surname, EmailAddress, Department,
                    @{Name="NomCanoniqueOU"; Expression={($_.DistinguishedName -split
"(?<=^.*,OU=)")[1] -replace "(?<=^.*,OU=)(.*)", '$1'}}

# Exporter les résultats au format CSV
$utilisateurs | Export-Csv -Path $cheminSortie -NoTypeInformation
```

Suppression user CSV

```
# Importer le contenu du fichier CSV
$csvPath = "chemin_vers_le_fichier\fichier.csv"
$csvData = Import-Csv -Path $csvPath

# Parcourir chaque ligne du fichier CSV
foreach ($row in $csvData) {
    $logon = $row.logon

    # Vérifier si le compte existe dans Active Directory
    if (Get-ADUser -Filter "SamAccountName -eq '$logon'") {
        # Supprimer le compte Active Directory
        Remove-ADUser -Identity $logon -Confirm:$false
        Write-Host "Le compte $logon a été supprimé."
    } else {
        Write-Host "Le compte $logon n'a pas été trouvé dans Active Directory."
    }
}
```

Extract nom de dossier

```
# Get the folder names
$directory = "C:\Path\to\Directory"
$folderNames = Get-ChildItem -Path $directory -Directory | Select-Object -ExpandProperty Name

# Specify the output file path
$outputFilename = "C:\Path\to\output.csv"

# Export the folder names to CSV
$folderNames | Export-Csv -Path $outputFilename -NoTypeInformation

Write-Host "Folder names exported to $outputFilename."
```

Crypt password + line to add on script

Nous devons d'abord transformer notre mot de passe en SecureString pour que celui-ci ne soit pas utilisé en clair

```
ConvertTo-SecureString -String "motdepasse" -AsPlainText -Force | ConvertFrom-SecureString
```

```
PS C:\Users\thoma> ConvertTo-SecureString -String "motdepasse" -AsPlainText -Force | ConvertFrom-SecureString
01000000d08c9ddf0115d1118c7a00c04fc297eb01000000a70604f6ea2b40469d5266fd66e8ea4a000000000200000000001066000000010000200000030ad8e063c5fbb620766fd9e904611aab26391f20fb047d8d0f487ed4362cccb000000000e8000000020000200000000df5055deacbe90efa1973bb78e144a4a4e888db9865bd14e6b351dcd0b0e8fa2200000008041d82649e1d322c14f1d43caf6a6c9296dce3213b295b7b19e7e88e182578b5400000007e5c01dec1ebc0fc93c7c9e5c5e2e70c53879348f8f9d13985d7914bcfea37a707c492dbc87ee40b43148424ede5d183baff76ac8a717417fbed8ac22dcb733e
```

Une fois la clé obtenue, vous pouvez la mettre dans le fichier qui servira pour votre script et ajouter les lignes ci dessous pour l'executer

```
$cryptPass = get-content "$(split-path $Script:MyInvocation.Mycommand.Path)\nomdufichier.txt" # cette commande permet d'executer le nom du fichier à lire, qui est supposé être dans le même répertoire que le script.
$SecPass = ConvertTo-SecureString -String $cryptPass #cette ligne permet de transformer en objet powershell le mdp Crypte
```

Verifier le issuer du certificat

```
$r = [Net.HttpWebRequest]::Create("https://enrollment.manage.microsoft.com/")
$r.GetResponse() | Out-Null
$r.ServicePoint.Certificate.Issuer
```

Le résultat attendu : un émetteur **Microsoft**. Si l'on voyait la CA interne, l'exception ne s'appliquerait pas.

