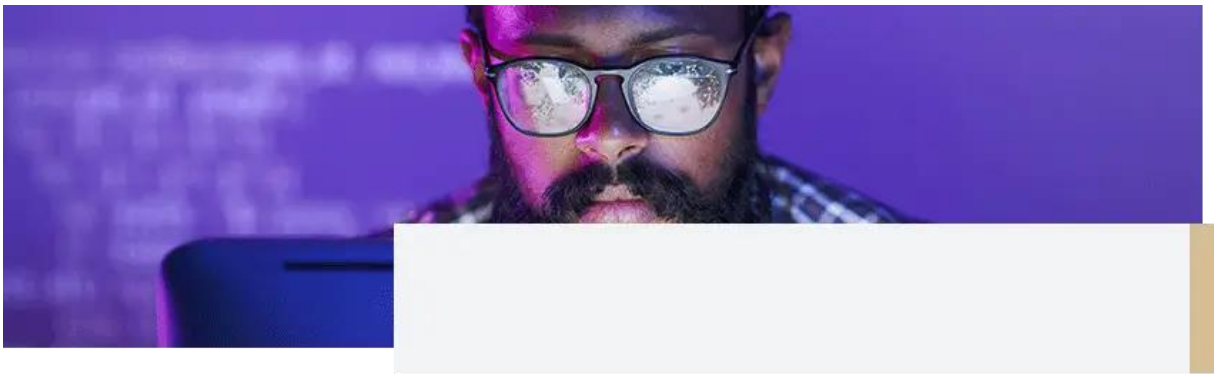
Script PowerShell : pour les nouveaux arrivants.



Ce script a été créé dans le cadre d'un projet que **V2V** m'a confié pour améliorer la sécurité et la gestion des identifiants lors du processus de recrutement. Initialement, lors du recrutement, les nouveaux arrivants recevaient une feuille papier avec un QR code qu'ils devaient scanner pour accéder à un formulaire. Une fois le formulaire rempli, ils obtenaient leurs identifiants pour accéder à l'entreprise.

Le but de ce projet est de **remplacer l'usage des feuilles de papier** par une solution numérique plus écologique et sécurisée. L'idée est d'utiliser **l'écran de verrouillage** de l'ordinateur pour afficher le QR code, permettant aux nouveaux arrivants de scanner le code et remplir le formulaire directement sur leur appareil.

Voici comment fonctionne le script dans ce cadre :

1. **Initialisation des paramètres** : Le script commence par définir l'emplacement du QR code à afficher sur l'écran de verrouillage et où sauvegarder l'image actuelle de fond d'écran. Il s'assure aussi que le script ne s'exécute qu'une seule fois par ordinateur.
2. **Vérification de l'exécution précédente** : Avant d'appliquer les modifications, le script vérifie si un fichier marqueur existe, ce qui permet d'éviter une exécution multiple.
3. **Sauvegarde de l'image actuelle** : Le script sauvegarde l'image actuelle du fond d'écran afin de pouvoir la restaurer si nécessaire.
4. **Personnalisation de l'écran de verrouillage** : Il remplace l'image de verrouillage par celle contenant le QR code. Cette image permet aux nouveaux arrivants de scanner le code et d'accéder au formulaire RH pour remplir leurs informations.
5. **Interaction avec l'utilisateur** : Une fois que le QR code est affiché, le script attend que l'utilisateur déverrouille l'écran pour s'assurer qu'il scanne le QR code et remplit le formulaire.
6. **Rétablissement des paramètres** : Après l'exécution du script, l'image de verrouillage peut être réinitialisée à l'image par défaut.

Ce projet permet à **V2V** d'éliminer l'utilisation de papier tout en garantissant un processus de recrutement fluide et sécurisé. Les identifiants sont remis directement après que le formulaire ait été rempli via le QR code, ce qui améliore la sécurité des données et simplifie le processus d'intégration des nouveaux arrivants.

1. Initialisation des paramètres

```
# Forcer la mise à jour des stratégies de groupe sur l'ordinateur local
Invoke-GPUpdate -RandomDelayInMinutes 0 -Force

# Définir le chemin de l'image actuelle et de sauvegarde
$currentImagePath =
"\\lov.local\SYSVOL\lov.local\scripts\wallpaper\img_actuels\WALLPAPER_V2V_Q
R_CODE.png"
$backupPath = "C:\Windows\Web\Sauvegardes\WALLPAPER_V2V_QR_CODE.png"
$scriptMarker = "C:\Windows\Web\Sauvegardes\script_executed.marker"
```

Explication :

- Le script commence par forcer la mise à jour des stratégies de groupe locales (`Invoke-GPUpdate`), ce qui peut être utile si des changements dans les stratégies de groupe sont nécessaires.
- Ensuite, il définit le chemin de l'image actuelle du fond d'écran à utiliser, ainsi que le chemin pour sauvegarder cette image localement (`$backupPath`) et l'emplacement d'un fichier marqueur (`$scriptMarker`), utilisé pour s'assurer que le script ne soit exécuté qu'une seule fois.

---

2. Vérification si le script a déjà été exécuté

```
# Vérifier si le script a déjà été exécuté en vérifiant l'existence d'un
marqueur
if (Test-Path $scriptMarker) {
    Write-Host "Le script a déjà été exécuté. Le script va se terminer."
    exit
}
```

Explication :

- Avant de poursuivre, le script vérifie si le fichier marqueur (`script_executed.marker`) existe. Si ce fichier est présent, cela signifie que le script a déjà été exécuté, et dans ce cas, il s'arrête immédiatement (`exit`).

---

3. Création de sauvegardes

```
# Créer le dossier de sauvegarde s'il n'existe pas déjà
$saveFolderPath = "C:\Windows\Web\Sauvegardes"
if (-not (Test-Path $saveFolderPath)) {
    New-Item -ItemType Directory -Path $saveFolderPath
}

# Sauvegarder l'image actuelle localement sur l'ordinateur
if (-not (Test-Path $backupPath)) {
    Copy-Item -Path $currentImagePath -Destination $backupPath -Force
}
```

```
# Créer un marqueur pour indiquer que le script a été exécuté
New-Item -Path $scriptMarker -ItemType File
```

Explication :

- Le script vérifie si un dossier de sauvegarde existe (`C:\Windows\Web\Sauvegardes`). S'il n'existe pas, il est créé.
- L'image actuelle (celle définie par `$currentImagePath`) est ensuite copiée dans le dossier de sauvegarde si elle n'y est pas déjà présente.
- Enfin, un fichier marqueur (`script_executed.marker`) est créé pour indiquer que le script a été exécuté avec succès.

---

4. Personnalisation de l'écran de verrouillage

```
# Modifier l'écran de verrouillage avec l'image personnalisée
Set-ItemProperty -Path
'HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Personalizati
onCSP' -Name 'LockScreenImagePath' -Value $currentImagePath

# Actualiser les paramètres de bureau pour appliquer les modifications
RUNDLL32.EXE user32.dll, UpdatePerUserSystemParameters

# Attendre que l'utilisateur déverrouille l'écran
Write-Host "Votre écran est verrouillé. Appuyez sur une touche une fois que
vous l'avez déverrouillé."
Pause
```

Explication :

- L'image de verrouillage (par défaut) de Windows est modifiée via une entrée du registre à l'emplacement `PersonalizationCSP`. Le chemin de l'image est défini par `$currentImagePath`, pointant vers l'image QR Code.
- Le script actualise ensuite les paramètres de bureau pour appliquer la nouvelle image de verrouillage avec `RUNDLL32.EXE user32.dll, UpdatePerUserSystemParameters`.
- Enfin, le script attend que l'utilisateur déverrouille l'écran avant de poursuivre, avec un message invitant l'utilisateur à appuyer sur une touche après avoir déverrouillé l'écran.

---

5. Modification avec les images par défaut

```
# Définir les chemins des nouvelles images par défaut
$newDefaultImages = @(
    "C:\Windows\Web\Screen\img100.jpg",
    "C:\Windows\Web\Screen\img101.png",
    "C:\Windows\Web\Screen\img102.jpg",
    "C:\Windows\Web\Screen\img103.png",
    "C:\Windows\Web\Screen\img104.jpg",
    "C:\Windows\Web\Screen\img105.jpg"
)
```

```
# Modifier l'écran de verrouillage avec les images par défaut
foreach ($imagePath in $newDefaultImages) {
    # Modifier l'écran de verrouillage avec l'image par défaut
    Set-ItemProperty -Path
'HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\PersonalizationCSP' -Name
'LockScreenImagePath' -Value $imagePath

    # Actualiser les paramètres de bureau pour appliquer les modifications
    RUNDLL32.EXE user32.dll, UpdatePerUserSystemParameters

    # Attendre que l'utilisateur déverrouille l'écran
    Write-Host "Votre écran est verrouillé avec l'image $($imagePath).
Appuyez sur une touche une fois que vous l'avez déverrouillé."
    Pause
}
```

Explication :

- Le script définit un tableau contenant plusieurs images par défaut à utiliser comme fond d'écran de verrouillage.
- Pour chaque image, le script applique l'image à l'écran de verrouillage, met à jour les paramètres, puis attend que l'utilisateur déverrouille l'écran avant de passer à l'image suivante.
- Cela permet de personnaliser successivement l'écran de verrouillage avec une série d'images.

---

6. Définition et vérification des clés de registre

```
# Définir le chemin de la clé de registre
$keyPath =
'HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Personalizati
onCSP'

# Créer la clé de registre
New-Item -Path $registryPath -Force

# Vérifier si la clé de registre existe déjà, sinon la créer
if (-not (Test-Path $keyPath)) {
    New-Item -Path $keyPath -ItemType Directory | Out-Null
}

# Définir une valeur dans la clé de registre pour l'écran avant la session
Set-ItemProperty -Path $keyPath -Name 'OEMBackground' -Value 1
Set-ItemProperty -Path $keyPath -Name 'OEMBackgroundImage' -Value
$currentImagePath
```

Explication :

- Le script crée une clé de registre sous `PersonalizationCSP` si elle n'existe pas déjà, afin de configurer l'écran de verrouillage et le fond d'écran avant la session (avant la connexion).
- Il définit ensuite des valeurs dans cette clé de registre pour activer l'arrière-plan OEM (`OEMBackground`) et pour spécifier l'image à utiliser (`OEMBackgroundImage`).

7. Vérification et ajout de clés de registre

```
# Vérifier et ajouter la clé de registre 'OEMBackground'
if (-not (Test-Path "Registry::$keyPath\\OEMBackground")) {
    New-ItemProperty -Path $keyPath -Name 'OEMBackground' -Value 1
}

# Vérifier et ajouter la clé de registre 'OEMBackgroundImage'
if (-not (Test-Path "Registry::$keyPath\\OEMBackgroundImage")) {
    New-ItemProperty -Path $keyPath -Name 'OEMBackgroundImage' -Value
$currentImagePath
}

Write-Host "Clés de registre vérifiées et ajoutées si nécessaire."
```

Explication :

- Le script vérifie si les propriétés `OEMBackground` et `OEMBackgroundImage` existent déjà dans le registre, et si elles n'existent pas, il les crée.
- Cela permet de garantir que l'image du fond d'écran avant la session soit définie correctement pour les futures connexions.

---

8. Désactivation du fond d'écran personnalisé pour les prochaines connexions

```
# Vérifier si le script a déjà été exécuté lors de l'ouverture de session
en vérifiant l'existence d'un marqueur
if (Test-Path $scriptMarker) {
    # Supprimer la clé de registre pour empêcher le changement de fond
d'écran lors de la prochaine ouverture de session
    Remove-ItemProperty -Path $keyPath -Name 'OEMBackground'
    Remove-ItemProperty -Path $keyPath -Name 'OEMBackgroundImage'
    Write-Host "Fond d'écran personnalisé désactivé pour les prochaines
connexions."
    exit
}
```

Explication :

- Si le script est exécuté à nouveau lors de l'ouverture de session, il supprime les clés de registre définies précédemment pour désactiver le fond d'écran personnalisé.
- Cela garantit que la personnalisation ne sera appliquée qu'une seule fois.
