**Settings Restoration Script**

**Overview**

This script is designed to restore browser settings for Google Chrome and Microsoft Edge from backup directories located on a network share. It handles multiple user profiles and performs the following tasks:

1. Validates the hostname to ensure it is correctly formatted.
2. Logs operations and errors.
3. Handles network interruptions during file operations.
4. Checks backup directories for corruption.
5. Resets permissions for user profiles.
6. Restores specified files and folders for each user.

**Prerequisites**

* PowerShell 5.0 or higher.
* Administrative privileges on the workstation.
* Access permissions to backup directories and network shares.

**Execution Instructions**

**Prerequisites**

1. **PowerShell Version**: Ensure that the script is executed using PowerShell 5.0 or higher.
2. **Permissions**: The script requires administrative privileges on the workstation to perform file operations and reset permissions.
3. **Network Access**: Ensure access to the network share where browser settings backups are stored.

**Notes**

* **Hostname Verification**: The script verifies the hostname to ensure it starts with a four-digit number (^\d{4}). If the hostname does not match this pattern, the script will exit, assuming it is either incorrectly named or not a back office workstation.
* **Logging**: Operations and errors are logged using the LogMessage function, ensuring comprehensive tracking of script activities. Review ${hostname}\_Backup.log for details on each operation's success or failure.
* **Network Interruptions**: The script includes resilience against network interruptions during file operations through the HandleNetworkInterruptions function. It retries operations up to three times ($retryCount = 3) before logging and continuing.
* **Backup Directory Validation**: Before proceeding with file operations, the script checks the accessibility and integrity of the backup base directory \\${siteNumber}-pcName\Redacted\Browser Settings Backups\${hostname}. If this directory is inaccessible or corrupted, the script exits with an error message.
* **User Profile Handling**: For each user identified in the backup directory, the script attempts to restore browser settings (RestoreBrowserSettings) and reset permissions (ResetPermissions) for both Google Chrome and Microsoft Edge profiles.
* **Disk Space Check**: Before restoring browser settings, the script checks available disk space (C: drive) to ensure it meets the requirements ($requiredSpace). If insufficient space is detected, the script logs this and skips restoration operations for the affected user.
* **Results Summary**: Upon completion, the script generates a summary of successfully restored and failed operations for Chrome and Edge settings, as well as permission resets. These results are logged and displayed in the PowerShell console.

**Detailed Code Breakdown:**

**1. Set Global Variables**

The script initializes global variables including the hostname and site number, and sets the name of the log file.

$hostname = hostname

if ($hostname.Substring(0, 4) -match '^\d{4}') {

$siteNumber = $hostname.Substring(0, 4)

} else {

Write-Error "This computer, '${hostname}', is either named incorrectly or is not a back office workstation. Exiting script..."

Start-Sleep -Seconds 10

Exit

}

$logFileName = "${hostname}\_Backup.log"

$scriptName = [System.IO.Path]::GetFileName($MyInvocation.ScriptName)

**2. Function to Log Messages**

A function to log messages with a timestamp.

function LogMessage {

param (

[string]$Message

)

$TimeStamp = Get-Date -Format "yyyy-MM-dd HH:mm:ss"

try {

Add-Content -Path $logFileName -Value "$TimeStamp - $Message"

} catch {

Write-Error "$scriptName failed to write $Message to $logFileName on $hostname. Error: $\_"

}

}

**3. Function to Handle Network Interruptions**

A function to retry operations in case of network interruptions.

function HandleNetworkInterruptions {

param (

[ScriptBlock]$Operation

)

$retryCount = 3

for ($i = 1; $i -le $retryCount; $i++) {

try {

& $Operation

LogMessage "Operation succeeded on attempt $i."

return $true

} catch [System.IO.IOException] {

LogMessage "Network interruption occurred. Retrying operation ($i/$retryCount)..."

Start-Sleep -Seconds 5

}

}

LogMessage "Operation failed after $retryCount retries due to network interruptions."

return $false

}

**4. Function to Check Backup Directory Corruption**

A function to check if a backup directory is corrupted or inaccessible.

function CheckBackupDirCorruption {

param (

[string]$BackupDir

)

if (!(Test-Path -Path $BackupDir -ErrorAction SilentlyContinue)) {

LogMessage "$'{BackupDir}' doesn't exist, assumed corrupted" "ERROR"

return $true

}

try {

Get-ChildItem $BackupDir -ErrorAction Stop | Out-Null

LogMessage "'${BackupDir}' is intact and accessible"

return $false

} catch {

LogMessage "'${BackupDir}' is corrupted or inaccessible" "ERROR"

return $true

}

}

**5. Function to Reset Permissions**

A function to reset permissions for user directories.

function ResetPermissions {

param (

[string]$User,

[string]$Path

)

try {

if (Test-Path -Path $Path) {

icacls $Path /reset /T | Out-Null

LogMessage "${User}'s permissions have been reset for: $Path"

return $true # Permission reset succeeded

} else {

LogMessage "Failed to reset permissions for ${User}, path does not exist: $Path" "ERROR"

return $false # Permission reset failed

}

} catch {

LogMessage "An error occurred while resetting ${User}'s permissions for: $Path. Error: $\_"

return $false # Permission reset failed

}

}

**6. Function to Restore Browser Settings**

A function to restore browser settings from backup directories.

function RestoreBrowserSettings {

param (

[string]$User,

[string]$ProfileDir,

[string]$BackupDir,

[string[]]$FilesToRestore,

[string[]]$FoldersToRestore

)

try {

if (Test-Path -Path $ProfileDir) {

foreach ($file in $FilesToRestore) {

$backupPath = Join-Path -Path $BackupDir -ChildPath $file

if (Test-Path -Path $backupPath) {

if (HandleNetworkInterruptions {

Copy-Item -Path $backupPath -Destination $ProfileDir -Recurse -Force

}) {

LogMessage "Restored file for ${User}: $file"

} else {

throw "Unstable network."

}

}

}

foreach ($folder in $FoldersToRestore) {

$backupPath = Join-Path -Path $BackupDir -ChildPath $folder

if (Test-Path -Path $backupPath) {

$destinationPath = Join-Path -Path $ProfileDir -ChildPath $folder

if (HandleNetworkInterruptions {

Copy-Item -Path $backupPath -Destination $destinationPath -Recurse -Force

}) {

LogMessage "Restored folder for ${User}: $folder"

} else {

throw "Unstable network."

}

}

}

return $true # Restoration succeeded

} else {

LogMessage "Profile directory does not exist for ${User}: $ProfileDir"

return $false # Restoration failed

}

} catch {

LogMessage "An error occurred while restoring ${User}'s browser settings from: $BackupDir to: $ProfileDir. Error: $\_"

return $false # Restoration failed

}

}

**7. Main Script Execution**

The main execution section of the script.

try {

# Get list of user directories from backup location

$backupBaseDir = "\\${siteNumber}-pcName\Redacted\Browser Settings Backups\${hostname}"

# Check for backup directory accessibility

if (!(Test-Path -Path $backupBaseDir)) {

throw "Backup base directory does not exist or is inaccessible: $backupBaseDir"

}

# Check for backup directory corruption or inaccessibility

if (CheckBackupDirCorruption -BackupDir $backupBaseDir) {

throw "Backup base directory is corrupted or inaccessible: $backupBaseDir"

}

$userDirs = Get-ChildItem -Path $backupBaseDir -Directory

$userList = $userDirs.Name

# Check that $userList is not empty

if ($userList.Count -eq 0) {

throw "No user directories found in backup location: $backupBaseDir"

}

# Initialize lists to track successful and failed restorations for Chrome and Edge separately

$successfulChromeRestoration = @()

$failedChromeRestoration = @()

$successfulEdgeRestoration = @()

$failedEdgeRestoration = @()

$successfulChromePermissionReset = @()

$failedChromePermissionReset = @()

$successfulEdgePermissionReset = @()

$failedEdgePermissionReset = @()

$successfulTotalRestore = @()

# Iterate through each user to restore browser settings for Chrome and Edge separately

foreach ($user in $userList) {

$userProfileDir = "C:\Users\$user"

$userBackupDir = Join-Path -Path $backupBaseDir -ChildPath $user

# Check if the user profile directory exists

if (Test-Path -Path $userProfileDir) {

# Define profile directories for Chrome and Edge

$chromeProfileDir = Join-Path -Path $userProfileDir -ChildPath "AppData\Local\Google\Chrome\User Data\Default"

$edgeProfileDir = Join-Path -Path $userProfileDir -ChildPath "AppData\Local\Microsoft\Edge\User Data\Default"

# Define backup directories for Chrome and Edge

$chromeBackupDir = Join-Path -Path $userBackupDir -ChildPath "Chrome"

$edgeBackupDir = Join-Path -Path $userBackupDir -ChildPath "Edge"

# Check for sufficient disk space before proceeding

$requiredSpace = (Get-ChildItem -Path $chromeBackupDir -Recurse | Measure-Object -Property Length -Sum).Sum +

(Get-ChildItem -Path $edgeBackupDir -Recurse | Measure-Object -Property Length -Sum).Sum

$freeSpace = [System.IO.DriveInfo]::GetDriveInfo("C:").AvailableFreeSpace

if ($freeSpace -lt $requiredSpace) {

LogMessage "Insufficient disk space for user ${user}. Required: $requiredSpace, Available: $freeSpace"

$failedChromeRestoration += $user

$failedEdgeRestoration += $user

continue

}

# Restore Chrome settings and update lists accordingly

if (RestoreBrowserSettings -User $user -ProfileDir $chromeProfileDir -BackupDir $chromeBackupDir -FilesToRestore @("Bookmarks", "Preferences", "Login Data", "History", "Cookies", "Web Data") -FoldersToRestore @("Extensions", "Local Storage", "Session Storage", "Sync Data")) {

$successfulChromeRestoration += $user

} else {

$failedChromeRestoration += $user

}

# Restore Edge settings and update lists accordingly

if (RestoreBrowserSettings -User $user -ProfileDir $edgeProfileDir -BackupDir $edgeBackupDir -FilesToRestore @("Bookmarks", "Preferences", "Login Data", "History", "Cookies", "Web Data") -FoldersToRestore @("Extensions", "Local Storage", "Session Storage", "Sync Data")) {

$successfulEdgeRestoration += $user

} else {

$failedEdgeRestoration += $user

}

# Reset permissions for Chrome and update lists accordingly

if (ResetPermissions -User $user -Path $chromeProfileDir) {

$successfulChromePermissionReset += $user

} else {

$failedChromePermissionReset += $user

}

# Reset permissions for Edge and update lists accordingly

if (ResetPermissions -User $user -Path $edgeProfileDir) {

$successfulEdgePermissionReset += $user

} else {

$failedEdgePermissionReset += $user

}

} else {

LogMessage "User profile directory does not exist: ${userProfileDir}. Skipping user: ${user}."

$failedChromeRestoration += $user

$failedEdgeRestoration += $user

}

}

# Generate list of users that passed every step

# Combine all success and failure arrays

$allSuccessful = $successfulChromeRestoration + $successfulEdgeRestoration + $successfulChromePermissionReset + $successfulEdgePermissionReset

$allFailed = $failedChromeRestoration + $failedEdgeRestoration + $failedChromePermissionReset + $failedEdgePermissionReset

# Filter out users who have failed any of the steps

$successfulTotalRestore = $allSuccessful | Where-Object {

$user = $\_

$allFailed -notcontains $user

}

# Remove duplicates

$successfulTotalRestore = $successfulTotalRestore | Select-Object -Unique

#Log and display results

if ($successfulTotalRestore.Count -gt 0) {

$successMessage = "Successfully restored Chrome and Edge settings for: $($successfulTotalRestore -join ', ')."

LogMessage $successMessage

Write-Host $successMessage

}

if ($failedChromeRestoration.Count -gt 0) {

$failedRestoreMessage = "Failed to restore Chrome settings for: $($failedChromeRestoration -join ', ')."

LogMessage $failedRestoreMessage

Write-Host $failedRestoreMessage

}

if ($failedEdgeRestoration.Count -gt 0) {

$failedRestoreMessage = "Failed to restore Edge settings for: $($failedEdgeRestoration -join ', ')."

LogMessage $failedRestoreMessage

Write-Host $failedRestoreMessage

}

if ($failedChromePermissionReset.Count -gt 0) {

$failedResetMessage = "Failed to reset Chrome file permissions for: $($failedChromePermissionReset -join ', ')."

LogMessage $failedResetMessage

Write-Host $failedResetMessage

}

if ($failedEdgePermissionReset.Count -gt 0) {

$failedResetMessage = "Failed to reset Edge file permissions for: $($failedEdgePermissionReset -join ', ')."

LogMessage $failedResetMessage

Write-Host $failedResetMessage

}

} catch {

LogMessage "An error occurred: $\_"

Write-Host "An error occurred: $\_"

}

Start-Sleep -Seconds 10

Exit