LOSTE TRADI-FRANCE

Sécurisation des Scripts deSauvegarde PowerShell (DPAPI)

Scripts PowerShell commentésDPAPI · Robocopy · Journalisation

Rédacteur	Paul Barrault-Gautier
Entreprise	Loste Tradi-France
Poste	Administrateur Réseau & Systèmes
Date	Avril 2026
Version	1.0
Classification	Usage interne — Confidentiel

1. Architecture des scripts

Chaque script de sauvegarde Loste suit la même structure modulaire. Le script principal orchestre les trois composants : chargement sécurisé des credentials (DPAPI), exécution de la sauvegarde (Robocopy) et journalisation des résultats.

Fichier	Rôle
`Backup-Site.ps1`	Script principal — orchestre credential, Robocopy et log
`creds\nas-cred.xml`	Fichier credential chiffré DPAPI (généré à l'INIT)
`logs\backup-YYYYMMDD.log`	Journal d'exécution quotidien
`dashboard\status.json`	Fichier de statut pour le dashboard NAS centralisé

2. Script principal — Backup-Site.ps1

Le script ci-dessous est la version complète et commentée du script de sauvegarde sécurisé. Il s'applique à un site Loste donné.

```
#Requires -Version 5.1
# ============================================================
# Backup-Site.ps1
# Script de sauvegarde sécurisé — Loste Tradi-France
# Auteur  : Paul Barrault-Gautier
# Version : 2.0 — Avril 2026
# Dépend  : nas-cred.xml (généré par procédure INIT)
# ============================================================

# ─── PARAMÈTRES ──────────────────────────────────────────────
[CmdletBinding()]
param(
    [string]$SiteName    = 'Site-Production-1',
    [string]$SourcePath  = 'D:\Donnees',
    [string]$NASShare    = '\\nas-loste\backup\site-prod-1',
    [string]$CredPath    = 'C:\Scripts\Backup\creds\nas-cred.xml',
    [string]$LogDir      = 'C:\Scripts\Backup\logs',
    [string]$StatusFile  = '\\nas-loste\supervision\status.json',
    [int]   $RetentionDays = 30   # Jours de conservation des logs locaux
)

# ─── INITIALISATION ──────────────────────────────────────────
$Date      = Get-Date -Format 'yyyy-MM-dd'
$DateTime  = Get-Date -Format 'yyyy-MM-dd HH:mm:ss'
$LogFile   = Join-Path $LogDir "backup-$SiteName-$Date.log"
$ErrorFlag = $false

# Créer le dossier de logs s'il n'existe pas
if (-not (Test-Path $LogDir)) { New-Item -Path $LogDir -ItemType Directory | Out-Null
}

# ─── FONCTION DE JOURNALISATION ──────────────────────────────
function Write-Log {
    param(
        [string]$Message,
        [ValidateSet('INFO','WARN','ERROR','SUCCESS')]
        [string]$Level = 'INFO'
    )
    $Timestamp = Get-Date -Format 'HH:mm:ss'
    $Line = "[$Timestamp] [$Level] $Message"
```

```
    Add-Content -Path $LogFile -Value $Line -Encoding UTF8
    Write-Host $Line
    # Écrire également dans l'Observateur d'événements Windows
    $EntryType = switch ($Level) {
        'ERROR'   { 'Error' }
        'WARN'    { 'Warning' }
        default   { 'Information' }
    }
    Write-EventLog -LogName Application -Source 'Loste-Backup' \
        -EventId 2000 -EntryType $EntryType -Message $Message -ErrorAction
SilentlyContinue
}

# ── CHARGEMENT DES CREDENTIALS DPAPI ──────────────────────────
Write-Log "Demarrage sauvegarde $SiteName — Source : $SourcePath"

# Vérifier que le fichier credential existe
if (-not (Test-Path $CredPath)) {
    Write-Log "Fichier credential introuvable : $CredPath" 'ERROR'
    Write-Log "Executer la procedure INIT (DOC-PSB-01) pour generer le fichier."
'ERROR'
    exit 1
}

try {
    # Import-Clixml dechiffre automatiquement via DPAPI
    # Echoue si le compte ou le poste ne correspondent pas a l'INIT
    $NASCredential = Import-Clixml -Path $CredPath
    Write-Log "Credential NAS charge : $($NASCredential.UserName)"
}
catch {
    Write-Log "Echec dechiffrement DPAPI : $($_.Exception.Message)" 'ERROR'
    Write-Log "Verifier que le script tourne sous le bon compte de service." 'ERROR'
    exit 1
}

# ── MONTAGE DU PARTAGE NAS ────────────────────────────────────
$NetDrive = 'Z:'

# Supprimer un éventuel lecteur Z: résiduel
if (Test-Path $NetDrive) {
    net use $NetDrive /delete /y | Out-Null
}

try {
    # Monter le partage NAS avec les credentials DPAPI
    $NetCred = $NASCredential.GetNetworkCredential()
    net use $NetDrive $NASShare /user:$($NetCred.UserName) $($NetCred.Password) |
Out-Null
    Write-Log "Partage NAS monte : $NASShare -> $NetDrive"
}
catch {
    Write-Log "Echec montage NAS : $($_.Exception.Message)" 'ERROR'
    $ErrorFlag = $true
}

# ── EXÉCUTION DE LA SAUVEGARDE (ROBOCOPY) ─────────────────────
if (-not $ErrorFlag) {
    $RoboArgs = @(
        $SourcePath,          # Source
        "$NetDrive\",          # Destination (lecteur NAS)
        '/MIR',               # Miroir : copie + supprime les fichiers absents
```

```
        '/Z',                 # Mode redémarrable (reprend si coupure réseau)
        '/W:5',               # Attente 5s entre tentatives
        '/R:3',               # 3 tentatives avant d'abandonner un fichier
        '/LOG+:{0}' -f $LogFile,  # Ajouter la sortie Robocopy au log
        '/NP',                # Ne pas afficher la progression en %
        '/NDL',               # Ne pas lister les répertoires
        '/TEE'                # Afficher ET écrire dans le log
    )

    Write-Log "Lancement Robocopy : $SourcePath -> $NetDrive"
    $RoboProcess = Start-Process -FilePath 'robocopy.exe' \
        -ArgumentList $RoboArgs -Wait -PassThru -NoNewWindow

    # Codes de retour Robocopy :
    # 0 = Aucun fichier copié (déjà à jour)
    # 1 = Fichiers copiés avec succès
    # 2 = Fichiers supplémentaires détectés
    # 4 = Fichiers mal correspondants
    # 8+ = Erreurs de copie
    $RoboExit = $RoboProcess.ExitCode
    if ($RoboExit -ge 8) {
        Write-Log "Robocopy terminé avec des erreurs (code $RoboExit)" 'ERROR'
        $ErrorFlag = $true
    } else {
        Write-Log "Robocopy termine avec succes (code $RoboExit)" 'SUCCESS'
    }
}

# ── DÉMONTAGE DU PARTAGE NAS ──────────────────────────────
if (Test-Path $NetDrive) {
    net use $NetDrive /delete /y | Out-Null
    Write-Log "Partage NAS demonte."
}

# ── NETTOYAGE DES ANCIENS LOGS ────────────────────────────
Get-ChildItem -Path $LogDir -Filter '*.log' |
    Where-Object { $_.LastWriteTime -lt (Get-Date).AddDays(-$RetentionDays) } |
    Remove-Item -Force
Write-Log "Logs anterieurs a $RetentionDays jours supprimes."

# ── MISE À JOUR DU FICHIER DE STATUT (SUPERVISION NAS) ────
$Status = @{
    Site        = $SiteName
    Date        = $DateTime
    Status      = if ($ErrorFlag) { 'ERROR' } else { 'OK' }
    RoboCode    = $RoboExit
    LogFile     = $LogFile
}

try {
    # Lire le fichier JSON existant (contient les statuts de tous les sites)
    $AllStatus = if (Test-Path $StatusFile) {
        Get-Content $StatusFile -Raw | ConvertFrom-Json
    } else { @{} }

    # Mettre à jour le statut du site courant
    $AllStatus | Add-Member -MemberType NoteProperty -Name $SiteName \
        -Value $Status -Force
    $AllStatus | ConvertTo-Json -Depth 3 | Set-Content $StatusFile -Encoding UTF8
    Write-Log "Statut supervision mis a jour : $StatusFile"
}
```

```
catch {
    Write-Log "Impossible de mettre a jour le fichier supervision :
$($_.Exception.Message)" 'WARN'
}

# ─── CONCLUSION ──────────────────────────────────────────────
$FinalStatus = if ($ErrorFlag) { 'ECHEC' } else { 'SUCCES' }
Write-Log "=== Sauvegarde $SiteName terminee : $FinalStatus ==" $(if ($ErrorFlag)
{'ERROR'} else {'SUCCESS'})
exit $(if ($ErrorFlag) { 1 } else { 0 })
```