Fiche 03 : Premiers Scripts

1. Éditeurs PowerShell

Pour cette fiche, trois éditeurs gratuits ont été testés de manière très sommaire.

- PowerGUI script Editor de la société Quest Software (powergui.com),
- PowerShellAnalyser (www.powershellanalyzer.com),
- Graphical Windows PowerShell inclu dans la version Windows PowerShell V2 (CTP2).

Il en existe d'autres, voir : www.microsoft.com/technet/scriptcenter/topics/winpsh/toolbox.mspx
Ces éditeurs présentent au moins deux zones, une est la console de saisie des commandes PS (pour les tests) avec éventuellement une zone sortie pour afficher le résultat et une deuxième zone pour l'éditeur de scripts.
Dans les versions testées, il me semble que PowerGUI gère mieux la saisie automatique par tabulation (liste déroulante des commandes et des paramètres après la saisie du caractère "-"), insertion de "Snippet" (*if…else, foreach, etc..).*

2. Les fichiers scripts

Les fichiers de scripts Windows PowerShell ont l'extension **.ps1**. Pour exécuter ces scripts, il faut spécifier le chemin d'accès complet du fichier de script dans la console PS (l'extension est facultative).
Pour spécifier le répertoire actif, il faut utiliser le point (.)
Exemple :

`c:\test\testscript.ps1`	*ou :*	`c:\test\testscript`	*ou encore :*	`.\testscript.ps1`

La stratégie de sécurité de Windows PowerShell permet de déterminer si des scripts peuvent s'exécuter et s'ils doivent inclure une signature numérique (voir fiche N°11). Aucune stratégie d'exécution ne permet d'exécuter un script en double-cliquant sur l'icône du fichier .ps1.
Les éditeurs possèdent tous un bouton ou un raccourci pour lancer directement le script chargé.
Exemple, pour autoriser l'exécution de scripts non signés localement :

`Set-ExecutionPolicy remotesigned`	*ou pour avoir de l'aide :*	`Get-Help Set-ExecutionPolicy`

3. Un premier script

Une première version sans pipeline (on va dire classique):

```
1  # Premier script
2  # Recherche le mot "erreur" dans les fichiers log d'un dossier temp
3  # Affiche le nombre de fois où le mot est trouvé (un par ligne)
4  $dossier="c:\temp\*.log"
5  $recherche="erreur"
6  $i=0
7  $colFichier=Get-ChildItem $dossier
8  Foreach($fic in $colFichier){
9      $colLigne=Get-Content $fic
10     Foreach($ligne in $colLigne){
11         if ($ligne.Contains($recherche)) {
12             $i++
13         }
14     }
15 }
16 Write-Host  "nombre d'erreur(s) : $i"
```

Retourne une collection d'objets qui sont les fichiers .log du dossier c:\temp

Retourne un tableau qui contient toutes les lignes du fichier, ce tableau peut être parcouru comme une collection

Remarque : Dans la chaîne de caractères avec les guillemets doubles ("), $i est remplacé par sa valeur (pas avec les simples).

Une autre version avec pipeline, la commande de réception pouvant être située sur la ligne suivante :

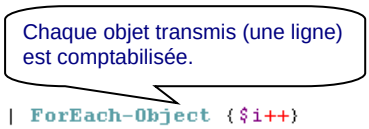
```
1  # Idem script01.ps1 mais avec pipeline
2  $dossier="c:\temp\*.log"
3  $recherche="erreur"
4  $i=0
5  Get-ChildItem $dossier | ForEach-Object {Get-Content $_} |
6  ForEach-Object {if ($_.contains($recherche)){$i++}}
7  Write-Host "nombre d'erreur(s) : $i"
```

Pour chaque objet transmis (un fichier), référencé par $_, extrait toutes les lignes pour les

Pour chaque objet transmis (une ligne), référencé par $_, utilise sa méthode contains (chaîne de

Une autre version avec la commande Select-String :

```powershell
# Idem script01.ps1 mais avec la commande Select-String
$dossier="c:\temp\*.log"
$recherche="erreur"
$i=0
Select-String -Path $dossier -Pattern $recherche -SimpleMatch | ForEach-Object {$i++}
Write-Host "nombre d'erreur(s) : $i"
```



Select-String extrait toutes les lignes des fichiers textes .log du dossier c:\temp comportant la chaîne recherchée.
SimpleMatch spécifie une simple chaîne pour la recherche (pas d'expression régulière).
Remarques : Get-Item peut remplacer ici Get-ChildItem, il serait judicieux de tester l'existence du dossier c:\temp avec Test-Path.