

INFORMATIQUE
TECHNIQUE



EXPERT

Windows PowerShell

Les fondamentaux du langage

Téléchargement
www.editions-eni.fr



Microsoft
Most Valuable
Professional

Arnaud
PETITJEAN

Robin LEMESLE

eni
Editions

Les éléments à télécharger sont disponibles à l'adresse suivante :
<http://www.editions-eni.fr>
Saisissez la référence ENI de l'ouvrage **EIPOWFOL** dans la zone de recherche
et validez. Cliquez sur le titre du livre puis sur le bouton de téléchargement.

Avant-propos

- 1. À propos de PowerShell. 17
- 2. À propos de cet ouvrage 18
- 3. À propos des auteurs 19

Introduction

- 1. Pourquoi utiliser les scripts ? 21
- 2. Historique des langages de script 22
- 3. Intérêt des scripts par rapport aux langages de programmation 25
- 4. Pour résumer... 25

Chapitre 1

À la découverte de PowerShell

- 1. Présentation de PowerShell. 27
- 2. Historique des versions 28
- 3. Plateformes supportées 30
 - 3.1 Plateformes client 30
 - 3.2 Plateformes serveur. 30
- 4. Prise en main 31
 - 4.1 Démarrage de la console PowerShell 31
 - 4.2 Découverte de la console classique 32
 - 4.3 L'environnement d'écriture de scripts intégré (ISE) 34
- 5. Une transition en douceur avec le passé 39

2 **Windows PowerShell**

Les fondamentaux du langage

6.	Système d'aide intégré	41
6.1	Mise à jour des fichiers d'aide	42
6.2	Configuration du système d'aide en entreprise	43
6.2.1	Dépôt de l'aide sur un partage réseau	43
6.2.2	Mise à jour de l'aide à partir d'un partage réseau	44
6.2.3	Forcer Update-Help à utiliser le dépôt	44
7.	Commandes de base	45
7.1	Constitution des commandes	45
7.2	Get-Command	46
7.3	Get-Help	49
7.3.1	Aide sur les commandes	50
7.3.2	Aide conceptuelle	52
7.4	Get-Member	53
8.	Gestion des répertoires et des fichiers	56
8.1	Get-ChildItem (alias : gci, ls, dir)	57
8.2	Set-Location (alias : sl, cd, chdir)	60
8.3	Get-Location (alias : gl, pwd)	60
8.4	New-Item (alias : ni, md)	61
8.4.1	Création d'un répertoire	61
8.4.2	Création d'un fichier	61
8.5	Remove-Item (alias : ri, rm, rmdir, rd, erase, del)	62
8.6	Move-Item (alias : mi, move, mv)	62
8.6.1	Déplacement de fichiers	63
8.6.2	Déplacement de répertoires	63
8.7	Rename-Item (alias : ren, rni)	64
8.7.1	Renommer un fichier	64
8.7.2	Renommer un dossier	64
8.8	Copy-Item (alias : cpi, cp, copy)	64
9.	Fournisseurs PowerShell	65

Chapitre 2
Manipulation des objets

- 1. Qu'est-ce que la notion d'objets ? 69
- 2. Manipulation des objets 72
 - 2.1 Les collections 72
 - 2.2 Différences de comportement entre versions de PowerShell 75
 - 2.3 Sélection/récupération des résultats 76
 - 2.3.1 Récupération des n premiers objets 77
 - 2.3.2 Récupération des n derniers objets 77
 - 2.3.3 Récupération des objets uniques 78
 - 2.3.4 Récupération d'une propriété particulière 78
 - 2.3.5 Sélection d'objets d'un tableau basée sur leur valeur d'index .. 80
 - 2.3.6 Examen de tous les objets d'une collection 81
 - 2.3.7 Regroupement d'objets 81
 - 2.3.8 Tri des objets 84
 - 2.3.9 Comptage/mesure des objets 85
 - 2.3.10 Comparaison d'objets 87
 - 2.4 Filtrer les objets 89
- 3. Formatage des objets pour l'affichage 91
 - 3.1 Format-List 92
 - 3.1.1 Affichage sélectif des propriétés d'un objet 93
 - 3.1.2 Affichage de toutes les propriétés disponibles d'un objet 94
 - 3.2 Format-Table 96
 - 3.2.1 Taille automatique d'un tableau 98
 - 3.2.2 Regroupement sur une propriété 99
- 4. Création d'objets personnalisés 99
 - 4.1 Transformation d'un objet existant 100
 - 4.2 Création d'un objet à partir de rien 102
 - 4.3 Ajout de membres 104
 - 4.3.1 Ajout d'une propriété 104
 - 4.3.2 Ajout d'une méthode 105
 - 4.4 Création d'une collection d'objets personnalisés 106

Chapitre 3 **Variables, constantes et types**

1. Les variables	109
1.1 Création et affectation	109
1.2 Déterminer le type d'une variable	110
1.3 Accéder au contenu d'une variable	110
2. Les constantes	110
3. Types de données.	111
4. Affectation manuelle de types et transtypage.	112
4.1 Conversion d'un nombre décimal en hexadécimal	114
4.2 Conversion d'un nombre décimal en octal (base 8)	114
4.3 Conversion d'un nombre décimal en binaire (base 2).	115
5. Rendre obligatoires la déclaration et l'initialisation des variables	115
6. Variables prédéfinies	116
6.1 Variables automatiques	116
6.2 Variables de préférence	120
7. Portée des variables	124
7.1 Portée globale (global:)	125
7.2 Portée locale (local:)	126
7.3 Portée script (script:)	126
7.4 Portée privée (private:)	127
7.5 Portée using (using:)	128
7.6 Portée workflow (workflow:)	128
8. Quantificateurs d'octets	128

Chapitre 4 **Opérateurs**

1. Introduction	131
1.1 Opérateurs arithmétiques.	131
2. Opérateurs de comparaison.	133
2.1 Comparaison sur des scalaires	133
2.2 Comparaison sur des tableaux.	134
3. Opérateurs de comparaison générique	135

4.	Opérateur de comparaison des expressions régulières	137
4.1	Opérations sur les scalaires	138
4.2	Opérations sur les tableaux	141
5.	Opérateur de plage	142
6.	Opérateurs d'appartenance	142
7.	Opérateur de remplacement	143
7.1	Remplacements à l'aide d'une expression régulière	144
7.2	Remplacements appliqués sur un tableau	144
8.	Opérateurs de type	146
9.	Opérateurs logiques	146
10.	Opérateurs binaires	147
11.	Opérateurs d'affectation	148
12.	Opérateurs de redirection	150
13.	Opérateurs de fractionnement et de concaténation	152
14.	Opérateur de format -f	153
14.1	Notions de base	153
14.2	Aller plus loin avec les formateurs de chaîne	154
15.	Récapitulatif sur les opérateurs	155

Chapitre 5 Tableaux

1.	Introduction	159
2.	Tableaux à une dimension	159
2.1	Initialiser un tableau vide	160
2.2	Initialiser un tableau avec des valeurs	161
2.3	Lire un tableau à une dimension	162
2.4	Concaténer deux tableaux	163
2.5	Ajouter un élément à un tableau	163
2.6	Modifier la valeur d'un élément	164
2.7	Supprimer un élément	164
2.8	Déterminer le nombre d'éléments d'un tableau	165
2.9	Convertir en chaîne le contenu d'un tableau	165
3.	Tableaux à plusieurs dimensions	166

6 --- Windows PowerShell

Les fondamentaux du langage

4. Tableaux associatifs	168
4.1 Tableaux associatifs standards	168
4.1.1 Déclarer un tableau associatif vide	168
4.1.2 Initialiser un tableau associatif avec des données	168
4.1.3 Ajout de données à un tableau associatif	169
4.1.4 Parcours d'un tableau associatif	170
4.2 Tableaux associatifs ordonnés	171

Chapitre 6 Boucles et conditions

1. Les boucles	173
1.1 Boucle While	173
1.2 Boucle Do-While	174
1.3 Boucle For	174
1.4 Boucle Foreach	175
1.4.1 Première technique	175
1.4.2 Seconde technique	176
2. Structure conditionnelle If, Else, Elseif	178
3. Switch	180

Chapitre 7 Fonctions et scripts

1. Fonctions	183
1.1 Anatomie d'une fonction	183
1.2 Utilisation des arguments	184
1.3 Utilisation des paramètres	185
1.4 Retour de valeurs	187
1.4.1 Retourner une valeur scalaire	187
1.4.2 Retourner un objet	187
1.5 Fonctions filtres	189
1.6 Introduction aux fonctions avancées	191
2. Scripts	193
2.1 Constitution d'un script	193
2.2 Commentaires	194

- 2.3 Exécution d'un script 194
- 2.4 La directive #Requires 195
- 2.5 Prise de conscience de l'environnement d'exécution (contexte) 196
- 2.6 Internationalisation 200
- 3. DotSourcing 202
- 4. Aide intégrée aux scripts et fonctions 203

Chapitre 8
Gestion des fichiers et des dates

- 1. La gestion de fichiers 209
 - 1.1 Envoi de données dans un fichier 210
 - 1.1.1 Fichiers textes avec Out-File 211
 - 1.1.2 Redirection du flux standard 213
 - 1.1.3 Création de fichiers binaires avec Set-Content 215
 - 1.2 Lecture de données avec Get-Content 220
 - 1.3 Recherche de contenu avec Select-String 225
 - 1.4 Gestion des fichiers CSV 230
 - 1.4.1 Import/export de données 230
 - 1.4.2 Conversion de données au format CSV 235
 - 1.4.3 Conversion de données à partir du format CSV 236
 - 1.5 Gestion des fichiers XML 238
 - 1.5.1 Chargement d'un fichier XML 239
 - 1.5.2 Gestion du contenu 239
 - 1.5.3 Sérialisation/désérialisation avec les commandes *-CliXML . 240
 - 1.6 Export de données en tant que page HTML 242
 - 1.7 Export de données avec Out-GridView 247
- 2. Dates 249
 - 2.1 Manipulation des objets DateTime 250
 - 2.2 Formatage des dates 253
 - 2.2.1 Formats standards 254
 - 2.2.2 Formats personnalisés 255
 - 2.3 Manipulation des dates 259
 - 2.3.1 Créer une date 259
 - 2.3.2 Modifier une date 259
 - 2.3.3 Comparer des dates 260

8 **Windows PowerShell**

Les fondamentaux du langage

2.3.4	Calculer un intervalle entre deux dates	260
2.3.5	Conversion d'une date exprimée en ticks.	262

Chapitre 9 **Profils PowerShell**

1.	Introduction	265
2.	Profils disponibles	266
3.	Ordre d'application des profils	268
4.	Création d'un profil	268
5.	Personnalisation de l'environnement	269
5.1	Modification du prompt	269
5.1.1	Un prompt haut en couleur.	271
5.1.2	Un prompt toujours à l'heure	271
5.2	Modification de la taille de la fenêtre	272
5.3	Modification des couleurs	273
5.4	Modification du titre de la fenêtre.	274
5.5	Ajout d'un message d'accueil personnalisé	275
5.6	Profil complet	276
6.	Exécuter PowerShell sans profil	277

Chapitre 10 **Modules et snap-ins**

1.	Introduction	279
2.	Les snap-ins	279
2.1	Lister les snap-ins installés	280
2.2	Importer un snap-in	281
2.3	Lister les commandes d'un snap-in	282
2.4	Décharger un snap-in	283
3.	Les modules	283
3.1	Installer un module.	284
3.2	Lister et importer les modules	285
3.2.1	Préfixer les commandes d'un module	292
3.3	Lister les commandes d'un module	293

- 3.4 Tracer l'utilisation des modules 293
- 3.5 Décharger un module 296

Chapitre 11
Gestion des erreurs

- 1. Introduction à la gestion des erreurs et au débogage 297
- 2. La gestion des erreurs 298
- 3. Les erreurs non critiques 298
 - 3.1 Variable de préférence : \$ErrorActionPreference 298
 - 3.2 Le paramètre -ErrorAction et les paramètres communs. 300
 - 3.3 Consignation des erreurs 303
 - 3.4 Le type ErrorRecord 304
 - 3.5 Redirection de l'affichage des messages d'erreur 306
 - 3.5.1 Redirection dans un fichier texte 307
 - 3.5.2 Redirection dans une variable 307
 - 3.5.3 Redirection des erreurs vers \$null 308
 - 3.6 Interception des erreurs non critiques. 308
 - 3.6.1 Cas général 308
 - 3.6.2 Cas des exécutables externes 308
- 4. Les erreurs critiques 309
 - 4.1 Interception des erreurs critiques avec trap 309
 - 4.2 Interception des erreurs critiques avec Try-Catch-Finally 317
 - 4.3 Déterminer le type des erreurs critiques 319
 - 4.4 Génération d'exceptions personnalisées 320
- 5. Le débogage 321
 - 5.1 Affichage de messages en mode verbose 321
 - 5.2 Affichage de messages en mode debug 322
 - 5.3 Affichage de messages en mode warning 322
 - 5.4 Forcer la définition des variables 323
 - 5.5 Exécution pas à pas 325
 - 5.5.1 Dans la console PowerShell classique 325
 - 5.5.2 Dans la console PowerShell ISE 330
 - 5.6 Mode trace de Set-PSDebug 331
 - 5.7 Trace-Command 334

Chapitre 12
Sécurité

1. La sécurité : pour qui ? Pourquoi ?	341
2. Les risques liés au scripting	341
3. Optimiser la sécurité PowerShell	342
3.1 La sécurité PowerShell par défaut	342
3.2 Les stratégies d'exécution	343
3.2.1 Les différentes stratégies d'exécution	343
3.2.2 Les étendues des stratégies d'exécution	346
3.2.3 Identifier la stratégie d'exécution courante	346
3.2.4 Appliquer une stratégie d'exécution	347
3.3 Scripts provenant d'Internet	349
3.4 Les Alternate Data Streams (ADS)	351
3.4.1 Les origines	351
3.4.2 Créer et lire les ADS	352
3.4.3 Observer et comprendre les ADS de vos fichiers .ps1	353
3.4.4 Modifier le ZoneId ou comment transformer un script distant en un script local	354
3.5 Chaînes sécurisées	355
3.5.1 Sécuriser une chaîne	356
3.5.2 Lire une chaîne sécurisée	359
3.6 Chiffrement	360
3.6.1 Chiffrer une chaîne	364
3.6.2 Déchiffrer une chaîne	366
3.7 Gestion des credentials	367
3.8 Demander la saisie d'un mot de passe de façon sécurisée	371
3.8.1 Utilisation de la commande Read-Host	371
3.8.2 Utilisation de la commande Get-Credential	371
4. Signature des scripts	372
4.1 Les signatures numériques	372
4.2 Les certificats	373
4.2.1 Acheter un certificat	373
4.2.2 Créer un certificat autosigné	373
4.3 Signer votre premier script	380
4.4 Exécuter des scripts signés	382

5. Gérer les stratégies d'exécution de PowerShell via les stratégies de groupe	383
--	-----

Chapitre 13 **Objets .NET**

1. Introduction à .NET	389
2. Le framework .NET	390
3. Utiliser des objets .NET avec PowerShell	391
3.1 Créer une instance de type (Objet)	394
3.2 Les assemblies	398
3.3 Charger une assembly	400
3.4 Lister les types contenus dans les assemblies	401
4. Tirer parti de la puissance de .NET	403
4.1 Wake-on-LAN	403
4.2 Compresser un fichier	404
4.3 Créer une bulle d'informations contextuelle (Balloon Tip)	406

Chapitre 14 **CIM / WMI**

1. Introduction	409
2. Des standards, encore des standards, mais pour quoi faire ?	410
2.1 Qu'est-ce que WMI ?	411
2.2 Qu'est-ce que CIM ?	411
2.3 CIM vs WMI	412
2.4 Et concrètement cela donne quoi ?	412
2.4.1 Configuration de serveurs DELL via iDRAC	412
2.4.2 Gestion de systèmes d'exploitation Linux depuis Windows ..	413
2.4.3 Gestion de Windows Server 2012/R2 depuis Linux	413
2.5 Difficultés à surmonter	413
3. Architecture générale et terminologie	414

4.	Commandes de la famille CIM	416
4.1	Jeu de commandes	416
4.2	Découverte des classes	417
4.2.1	Lister toutes les classes	418
4.2.2	Rechercher des classes contenant un mot particulier	418
4.3	Découverte des membres d'une classe	419
4.3.1	Lister les membres d'une classe	420
4.3.2	Rechercher des membres d'une classe	421
4.4	Récupération d'une ou plusieurs instances	422
4.5	Récupération d'une ou plusieurs instances avec filtre WQL/CQL	423
4.6	Invocation d'une méthode	425
5.	Commandes de la famille WMI	425
5.1	Recherche de classes et de membres	427
5.2	Récupération d'une ou plusieurs instances	428
6.	Établissement de sessions avec des machines distantes	430
6.1	Commande New-CimSession	431
6.2	Commande New-CimSessionOption	431
6.3	Commande Get-CimSession	432
6.4	Commande Remove-CimSession	432
7.	Monitoring de ressources avec la gestion des événements	433
7.1	Surveiller la création d'un processus local	433
7.2	Surveiller la création d'un processus à distance	435
7.3	Surveiller l'espace disque d'un serveur distant	438
7.4	Monitorer la suppression de fichiers	439
7.5	Quelques explications complémentaires	440
8.	Gestion basée sur les URI (Uniform Resource Identifier)	440
8.1	Anatomie d'un URI	441
8.2	Jeux de commandes PowerShell	442
8.2.1	Jeu de commandes de la famille WSMAN	442
8.2.2	Jeu de commandes de la famille CIM	443
8.3	Test de la bonne configuration d'un système	444
8.4	Envoi de requêtes CIM/WMI via un URI	444
8.4.1	Lister les services d'une machine distante	444
8.4.2	Déterminer la date d'installation d'une machine distante	445

- 9. Boîte à outils graphiques pour l'exploration de la base CIM/WMI 447
 - 9.1 Testeur WMI (Wbemtest.exe) 447
 - 9.2 CIM Studio 448
 - 9.3 SAPIEN WMI Explorer 2015 449

Chapitre 15
Exécution à distance

- 1. Introduction 451
- 2. Communications à distance du framework .NET 452
 - 2.1 Prérequis 453
 - 2.2 Déterminer les commandes à distance du framework .NET 453
 - 2.3 Le jeu de commandes 455
 - 2.4 Envoi de commandes à distance 456
- 3. Communications à distance Windows PowerShell 458
 - 3.1 Prérequis 459
 - 3.2 Configuration manuelle du service WinRM
 - dans un environnement Active Directory 461
 - 3.2.1 Activation du service WinRM 461
 - 3.2.2 Communiquer en HTTPS 465
 - 3.2.3 Changer les ports d'écoute 467
 - 3.2.4 Mécanismes d'authentification 469
 - 3.3 Configuration du service WinRM
 - dans un environnement Active Directory par GPO 471
 - 3.4 Configuration du service WinRM
 - dans un environnement Workgroup 474
 - 3.4.1 Configuration de la liste des machines de confiance
 - (trusted hosts list) 475
 - 3.4.2 Désactivation de l'UAC 476
 - 3.5 Problématique du "double saut" (notion de rebond) 476
 - 3.6 Gestion des configurations des sessions 478
 - 3.6.1 Généralités 478
 - 3.6.2 Configurations de session par défaut 480
 - 3.6.3 Modification des permissions 482
 - 3.6.4 Création d'une configuration de session personnalisée 483
 - 3.6.5 Création d'une configuration de session déléguée (RunAs) 491

3.7	Création d'une session à distance	493
3.8	Exécution de commandes à distance	496
3.9	Sessions WinRM en mode déconnecté	498
3.10	Exécution de scripts à distance	498
3.11	Ouverture d'une session interactive PowerShell à distance	503
3.11.1	Enter-PSSession	503
3.11.2	Powershell ISE (Integrated Scripting Environment)	506
3.12	Importation de commandes à distance	507

Chapitre 16

Études de cas

1.	Trouver les comptes d'ordinateurs périmés dans AD DS	511
1.1	Problématique	511
1.2	Difficultés à surmonter	512
1.3	Solution	512
2.	Lister les comptes d'utilisateurs inactifs dans AD DS	514
2.1	Problématique	514
2.2	Solution : faire du ménage !	514
3.	Changer le mot de passe Administrateur local à distance	518
3.1	Problématique	518
3.2	Difficultés à surmonter	519
3.3	Solution 1 : DCOM/RPC	519
3.4	Solution 2 : WSMAN/WinRM	521
4.	Surveiller l'arrivée d'un événement dans le journal	522
4.1	Problématique	522
4.2	Solution	522
5.	Créer des comptes utilisateurs par lot	525
5.1	Problématique	526
5.2	Solution	526
6.	Vérifier la version logicielle d'une application à distance	529
6.1	Problématique	529
6.2	Solution	530
7.	Mettre à jour la configuration réseau d'un ensemble de machines	532
7.1	Problématique	532

- 7.2 Solution 533
- 7.3 Test de la solution 535
- 8. Trouver les certificats expirés 539
 - 8.1 Problématique 539
 - 8.2 Solution 1 : Job PowerShell planifié local 539
 - 8.3 Solution 2 : Interrogation depuis un point central 543
- 9. Déléguer la gestion d'un serveur (quelques commandes seulement) 544
 - 9.1 Problématique 544
 - 9.2 Solution 545

Chapitre 17
Ressources complémentaires

- 1. Ressources Web 551
 - 1.1 Sites Internet francophones 551
 - 1.1.1 PowerShell-Scripting.com :
la communauté PowerShell francophone 551
 - 1.1.2 via PowerShell 553
 - 1.2 Sites Internet anglophones 554
 - 1.2.1 Windows PowerShell Blog 554
 - 1.2.2 PowerShell Magazine 555
 - 1.2.3 PowerShell.org 556
 - 1.2.4 PowerShell.com 557
 - 1.2.5 Get-Scripting 558
 - 1.2.6 CodePlex/GitHub 559
- 2. Outils tiers 561
 - 2.1 PowerGUI 561
 - 2.2 PowerShell Plus 562
 - 2.3 PowerShell Studio 2015 563
 - 2.4 PowerGadget 564
 - 2.5 ISE Steroids 564

16 _____ Windows PowerShell

Les fondamentaux du langage

Conclusion

1. Conclusion	565
-------------------------	-----

Annexes

1. Liste des commandes PowerShell v3 sur Windows Server 2012	567
2. Liste des alias	571
3. Liste des modules Windows Server 2012	574
4. Liste des modules Windows 8	575
5. Liste des sources de trace (Get-TraceSource)	575
6. Syntaxe des expressions régulières	576
7. Liste des verbes approuvés (Get-Verb)	579

Index	583
-----------------	-----

Chapitre 12 Sécurité

1. La sécurité : pour qui ? Pourquoi ?

L'arrivée des réseaux locaux et d'Internet a changé beaucoup de choses dans la manière de protéger son PC. Il ne suffit plus d'attacher son disque dur au radiateur et de fermer la porte du bureau le soir pour ne pas se faire voler ou pirater des données. Maintenant, protéger son poste de travail est devenu essentiel pour ne pas faire les frais d'intrusions ou de malveillances.

Mais alors contre qui se prémunir ? Eh bien, contre tout ce qui bouge... et même ce qui ne bouge pas. En effet, que ce soient des programmes malveillants, des utilisateurs mal intentionnés, voire des utilisateurs inexpérimentés, tous peuvent être considérés comme une menace. C'est pour cela que vous devez verrouiller votre système en établissant des règles de sécurité, en les appliquant et en vous assurant que les autres en font tout autant.

2. Les risques liés au scripting

Vous allez vite deviner que ce qui fait la force du scripting en fait aussi sa faiblesse. La facilité avec laquelle vous pouvez tout faire, soit en cliquant sur un script, soit en exécutant depuis la fenêtre de commande, peut vous mettre dans l'embarras si vous ne faites pas attention.

Imaginez un script de logon qui dès l'ouverture de la session la verrouille aussitôt ! Alors oui, c'est sympa entre copains, mais en entreprise, nous doutons que cela soit de bon ton. Plus grave encore, un script provenant d'une personne mal intentionnée ou vraiment peu expérimentée en PowerShell (dans ce cas, nous vous conseillons de lui acheter un exemplaire de ce livre...) peut parfaitement vous bloquer des comptes utilisateurs dans Active Directory, vous formater un disque, vous faire rebooter sans cesse. Enfin, vous l'avez compris, un script peut tout faire. Car même si aujourd'hui des alertes sont remontées jusqu'à l'utilisateur pour le prévenir de l'exécution d'un script, elles ne sont pas capables de déterminer à l'avance si un script est nuisible au bon fonctionnement du système.

Les risques liés au scripting se résument à une histoire de compromis : soit vous empêchez toute exécution de scripts, c'est-à-dire encourir le risque de vous « pourrir la vie » à faire et à refaire des tâches basiques et souvent ingrates, soit vous choisissez d'ouvrir votre système à PowerShell, en prenant soin de prendre les précautions qui s'imposent.

Mais ne vous laissez pas démoraliser car même si l'exécution de scripts vous expose à certains problèmes de sécurité, PowerShell se dote de certains concepts qui font de lui l'un des langages de script les plus sûrs. Il ne faut pas non plus oublier qu'en cas de problème de sécurité, c'est l'image tout entière de Microsoft qui en pâtit...

3. Optimiser la sécurité PowerShell

3.1 La sécurité PowerShell par défaut

Vous l'avez compris, la sécurité est une chose très importante, surtout dans le domaine du scripting. C'est pour cela que les créateurs de PowerShell ont inclus deux règles de sécurités par défaut.

Des fichiers ps1 associés au bloc-notes

L'extension « **.ps1** » des scripts PowerShell, est par défaut associée à l'éditeur de texte bloc-notes (ou Notepad). Ce procédé permet d'éviter de lancer des scripts potentiellement dangereux sur une mauvaise manipulation. Le bloc-notes est certes un éditeur un peu classique, mais a le double avantage d'être inoffensif et de ne pas bloquer l'exécution d'un script lorsque celui-ci est ouvert avec l'éditeur. Vous remarquerez cependant que l'édition (clic droit + **Modifier**) des fichiers **.ps1** est associée à l'éditeur ISE.

■ Remarque

Ce type de sécurité n'était pas mis en place avec les scripts VBS dont l'ouverture était directement associée au Windows Script Host.

Une stratégie d'exécution restreinte

La seconde barrière de sécurité est l'application de la stratégie d'exécution **Restricted** par défaut pour les postes clients et **RemoteSigned** celle par défaut pour les serveurs (depuis Windows Server 2012 R2).

La stratégie **Restricted** est la plus restrictive. C'est-à-dire qu'elle bloque systématiquement l'exécution de tout script. Seules les commandes tapées dans le shell seront exécutées. Pour remédier à l'inexécution des scripts, PowerShell requiert que l'utilisateur change le mode d'exécution avec la commande **Set-ExecutionPolicy** `<mode d'exécution>`. Mais pour ce faire il faut être administrateur de la machine.

■ Remarque

Peut-être comprenez-vous mieux pourquoi l'utilisation de PowerShell sur vos machines ne constitue pas un accroissement des risques, dans la mesure où certaines règles sont bien respectées.

3.2 Les stratégies d'exécution

PowerShell intègre un concept de sécurité que l'on appelle les stratégies d'exécution (execution policies) pour qu'un script non autorisé ne puisse pas s'exécuter à l'insu de l'utilisateur. Il existe sept configurations possibles : **Restricted**, **RemoteSigned**, **AllSigned**, **UnRestricted**, **Bypass**, **Default** et **Undefined**. À chacune d'elles correspond un niveau d'autorisation d'exécution de scripts particulier. Vous pourrez être amené à en changer en fonction de la stratégie que vous souhaitez appliquer.

3.2.1 Les différentes stratégies d'exécution

Restricted : c'est la stratégie la plus restrictive, et c'est aussi la stratégie par défaut sur les postes clients (de Windows XP à Windows 8.1). Elle ne permet pas l'exécution de scripts mais autorise uniquement les instructions en ligne de commande tapées dans la console (mode interactif). Cette stratégie peut être considérée comme la plus radicale étant donné qu'elle protège contre l'exécution involontaire des fichiers **.ps1**.

Lors d'une tentative d'exécution de script avec cette stratégie, un message de ce type est affiché dans la console :

```
■ .\Get-Script.ps1 : File C:\Temp\Script.ps1 cannot be loaded because running scripts is disabled on this system.
```

Si cette stratégie est celle définie par défaut lors de l'installation de PowerShell, il vous faudra la changer pour l'exécution de votre premier script.

Si vous essayez d'exécuter un script provenant d'Internet sans que celui-ci soit signé, le message suivant sera affiché dans la console.

```
.\Get-Script.ps1 : File C:\Temp\Script.ps1 cannot be loaded.  
The file C:\Temp\Script.ps1 is not digitally signed.
```

Remarque

Vous vous demandez sûrement comment PowerShell fait pour savoir que notre script provient d'Internet ? Réponse : Grâce aux « Alternate Data Streams » qui sont implémentés sous forme de flux cachés depuis des applications de communication telles que Microsoft Outlook, Internet Explorer, Outlook Express et Windows Messenger (voir partie traitant des Alternate Data Streams). En gros, lorsqu'un script est téléchargé à partir d'un client Microsoft, la provenance de celui-ci lui est attachée.

Unrestricted : avec cette stratégie, tout script, peu importe son origine, peut être exécuté sans demande de signature.

Cette stratégie affiche tout de même un avertissement lorsqu'un script téléchargé d'Internet tente d'être exécuté.

```
PS > .\script.ps1  
  
Security warning  
  
Run only scripts that you trust. While scripts from the internet can  
be useful, this script can potentially harm your computer. If you trust  
this script, use the Unblock-File cmdlet to allow the script to run  
without this warning message. Do you want to run C:\Temp\script.ps1?  
[D] Do not run [R] Run once [S] Suspend [?] Help (default is "D"):
```

Bypass : c'est la stratégie la moins contraignante, et par conséquent la moins sûre. Rien n'est bloqué et aucun message d'avertissement ne s'affiche. C'est donc la stratégie où le risque d'exécuter des scripts malveillants est le plus élevé.

Undefined : pas de stratégie d'exécution définie dans l'étendue courante. Si toutes les stratégies d'exécution de toutes les étendues sont non définies alors la stratégie effective appliquée sera la stratégie **Restricted**.

Default : positionne la stratégie par défaut, à savoir **Restricted**.

Remarque

Microsoft a mis en place ces mécanismes afin de tenter de limiter les risques liés à l'exécution de scripts provenant de l'extérieur de l'entreprise et donc potentiellement malveillants. La configuration par défaut permet d'atteindre cet objectif mais elle ne garantit en aucun cas une sécurité parfaite.

3.2.2 Les étendues des stratégies d'exécution

PowerShell permet de gérer l'étendue des stratégies. L'ordre d'application est le suivant :

- **Étendue Process** : la stratégie d'exécution n'affecte que la session courante (processus Windows PowerShell). La valeur affectée à l'étendue **Process** est stockée en mémoire uniquement ; elle n'est donc pas conservée lors de la fermeture de la session PowerShell.
- **Étendue CurrentUser** : la stratégie d'exécution appliquée à l'étendue **CurrentUser** n'affecte que l'utilisateur courant. Le type de stratégie est stocké de façon permanente dans la partie du registre **HKEY_CURRENT_USER**.
- **Étendue LocalMachine** : la stratégie d'exécution appliquée à l'étendue **LocalMachine** affecte tous les utilisateurs de la machine. Le type de stratégie est stocké de façon permanente dans la partie du registre **HKEY_LOCAL_MACHINE**.

La stratégie ayant une priorité 1 est plus propriétaire que celle ayant une priorité 3. Par conséquent, si l'étendue **LocalMachine** est plus restrictive que l'étendue **Process**, la stratégie qui s'appliquera sera quand même la stratégie de l'étendue **Process**. À moins que cette dernière soit de type **Undefined** auquel cas PowerShell appliquera la stratégie de l'étendue **CurrentUser** puis tentera d'appliquer la stratégie **LocalMachine**.

À noter que l'étendue **LocalMachine** est celle par défaut lorsque l'on applique une stratégie d'exécution sans préciser d'étendue particulière.

3.2.3 Identifier la stratégie d'exécution courante

La stratégie d'exécution courante s'obtient avec la commande `Get-ExecutionPolicy`.

Exemple

```
PS > Get-ExecutionPolicy
Restricted
```

Avec cette commande, nous bénéficions du commutateur `-List`. Grâce à lui, nous allons savoir quelles stratégies s'appliquent à nos étendues.

Par exemple

```
PS > Get-ExecutionPolicy -List

Scope ExecutionPolicy
-----
MachinePolicy          Undefined
UserPolicy             Undefined
Process               Undefined
CurrentUser            AllSigned
LocalMachine           Restricted
```