

Chapitre 5

La 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 malversations.

Mais alors contre qui se prémunir ? Hé bien, contre tout ce qui bouge... et même ce qui ne bouge pas. En effet, que ce soit 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 vous assurant que les autres en font 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 l'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 script, 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 au scripting, 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 nouveaux concepts qui facilitent grandement cet aspect du scripting.

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.

■ 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 stratégie d'exécution « restricted » par défaut (cf. stratégies d'exécution).

Cette stratégie 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 de script, PowerShell requiert que l'utilisateur change le mode d'exécution avec la commande **Set-ExecutionPolicy <mode d'exécution>**.

■ Remarque

Peut-être comprenez-vous mieux pourquoi le déploiement 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 quatre configurations possibles : **Restricted**, **RemoteSigned**, **AllSigned** et **unrestricted**. Chacune d'elles correspond à un niveau d'autorisation d'exécution de scripts particulier, et vous pourrez être amenés à en changer en fonction de la stratégie que vous souhaitez appliquer.

3.2.1 Les différentes stratégies d'exécution

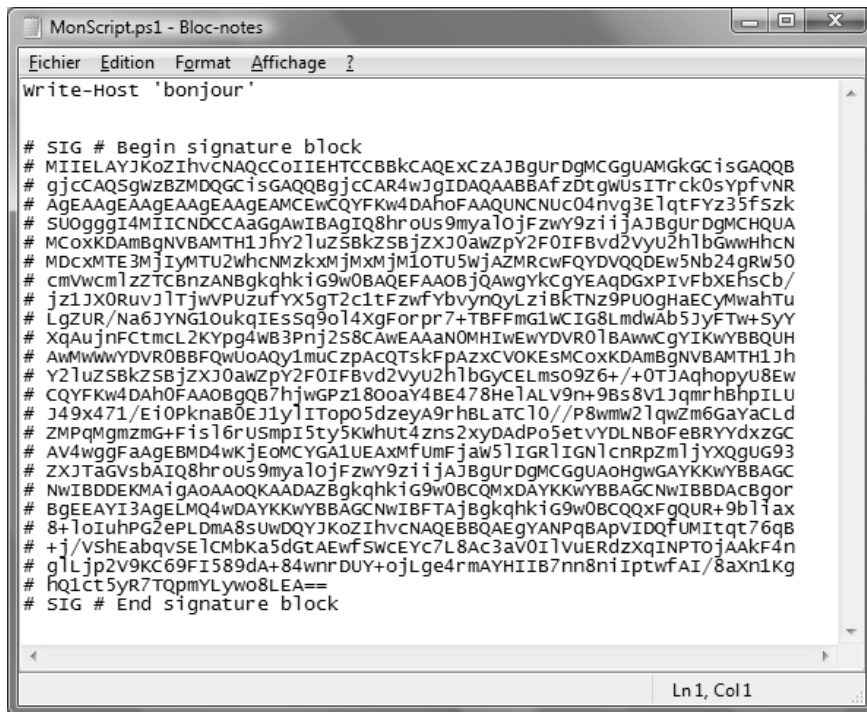
Restricted : c'est la stratégie la plus restrictive, elle ne permet pas l'exécution de script mais autorise uniquement les instructions en ligne de commande, c'est-à-dire uniquement dans le shell. Cette stratégie peut être considérée comme une stratégie radicale car elle protège l'exécution involontaire de fichiers « .ps1 », mais elle fait perdre l'énorme avantage qu'en est l'exécution.

Lors d'une tentative d'exécution de script avec cette stratégie, le message suivant est affiché dans la console :

```
Impossible de charger le fichier C:\Temp\list-group.ps1,  
car l'exécution de scripts est désactivée sur ce système.
```

À noter que cette stratégie est celle définie par défaut lors de l'installation de PowerShell. Il vous faudra donc la changer pour l'exécution de votre premier script.

AllSigned : c'est la stratégie permettant l'exécution de script la plus « sûre ». Elle autorise uniquement l'exécution des scripts signés, même si ceux-ci ont été créés localement. Un script signé est un script comportant une signature numérique comme celle présentée sur la figure suivante.



```

Fichier Edition Format Affichage ?
write-Host 'bonjour'

# SIG # Begin signature block
# MIEELAYJKoZIhvcNAQcCOIEHTCCBBkCAQEXCzaJBgurDgMCGGUAMGKGCisGAQQB
# gjcCAQSGwZBZMDQGCisGAQQBgjCCAR4wJgIDAQAABBAfzdtgwusITrck0sYpfvNR
# AgEAAgEAAGEAAGEAAGEAAGEAAGEAAGEAAGEAAGEAAGEAAGEAAGEAAGEAAGEAAGEA
# S0UgggI4MIICNCCAAGAwIBAgIQ8hr0us9mya1oJFzwY9ziijAJBgURDgMCHQUA
# MCOxKdAMBgnVBAMTH1Jhy21uz5BkZSBjZXJ0awZpY2F0IFBvd2Vyu2h1bGwwHhcn
# MDCxMTEm3jIyMTU2whcnMzkxMjMxMjM1OTU5wJAZMRCwFQYDVQQDEw5Nb24gRW50
# cmVwcm1zZTCBznANBgkqhkiG9w0BAQEFAAOBjQAwGyKCCgYEAqDgXPIvFBxehsCb/
# jz1JXORuvJlTjwVPuzufYX5gt2c1tFzwfYbvynqYLz1BkTNz9PUogHaECyMwahTu
# LgZUR/Na6JYNG1oukqIEsSq9o14XgForpr7+TBFFmG1wCIG8LmdwAb5JyFTw+syY
# XqAujnFctmCL2KYpg4wB3Pnj2S8CAwEAAaANOMHIwEwYDVR01BAwwCgYIKwYBBQUH
# AwMwwYDVR0BBFQwUoAQy1mucZpAcQTSkFPAZXCVOKEsMCOxKdAMBgnVBAMTH1Jh
# Y21uz5BkZSBjZXJ0awZpY2F0IFBvd2Vyu2h1bGyCELMs09Z6+/+0TJAqhopyu8Ew
# CQYFKw4DAh0FAAOBgQB7hJwGPz180oaY4BE478He1ALv9n+9BS8V1JqmrhBhpILU
# J49x471/Ei0Pknab0Ej1y1ITopo5dzeYA9rHBLaTC10//P8wmmw21qwz6GaYaCLd
# ZMPqMgmzmG+Fis16rUSmpI5ty5Kwhut4zns2xyDadPo5etvYDLNBOfEBRYdxzGC
# AV4wggFaAgEBMD4wkjEOMCYGA1UEAXmfumFjAw5lIGRlIGNlcnRpm1jYXQGU93
# ZXJTaGvsbA1Q8hr0us9mya1oJFzwY9ziijAJBgURDgMCGGUAOHGwAYKKwYBBAGC
# NwIBDDEKMAiga0AAOQKAADAZBgkqhkiG9w0BCQMxDAYKKwYBBAGCNwIBBDACBgor
# BgEeAYI3AgELMQ4wDAYKKwYBBAGCNwIBFTAjBgkqhkiG9w0BCQQxFgQUR+9b1iax
# 8+1oIuhPG2ePLDmA8sUwDQYJKoZIhvcNAQEBBQAEgYANPqBAPvIDQfUMItqt76qB
# +j/VshEabqvSElCMBka5dGtAEwFswCEYc7L8Ac3av0I1vuErdzXqINPTojAAKF4n
# g1Ljp2V9KC69FI589da+84wnrDUY+oJLge4rmAYHIIb7nn8niIptwFAI/8axn1Kg
# hQ1ct5yr7TQpmYLywo8LEA==
# SIG # End signature block

```

Exemple de script signé

Avec la stratégie **AllSigned**, l'exécution de scripts signés nécessite que vous soyez en possession des certificats correspondants (cf. partie sur la signature des scripts).

RemoteSigned : cette stratégie se rapporte à **AllSigned** à la différence près que seuls les scripts ayant une origine autre que locale nécessitent une signature. Par conséquent, cela signifie que tous vos scripts édités localement peuvent être exécutés sans être signés.

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

```
Impossible de charger le fichier C:\Temp\list-group.ps1.  
Le fichier C:\Temp \list-group.ps1 n'est pas signé numériquement.  
Le script ne sera pas exécuté sur le système.
```

■ Remarque

Vous vous demandez 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).

Unrestricted : c'est la stratégie la moins contraignante, et par conséquent la moins sécurisée. Avec elle, tout script, peu importe son origine, peut être exécuté sans demande de signature. C'est donc la stratégie où le risque d'exécuter des scripts malveillants est le plus élevé.

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

```
PS > .\list-group.ps1  
  
Avertissement de sécurité  
N'exécutez que des scripts que vous approuvez. Bien que les scripts  
en provenance d'Internet puissent être utiles, ce  
script est susceptible d'endommager votre ordinateur.  
Voulez-vous exécuter C:\Temp\list-group.ps1 ?  
[N] Ne pas exécuter [O] Exécuter une fois  
[S] Suspendre [?] Aide (la valeur par défaut est « N ») :
```

3.2.2 Appliquer une stratégie d'exécution

La plupart du temps, la stratégie **restricted** n'est pas adaptée, puisqu'elle ne permet pas l'exécution de script. Nous ne pouvons donc que vous conseiller d'en choisir une plus souple, de façon à pouvoir jouir des nombreux avantages qu'offre le scripting PowerShell.

Le changement de stratégie d'exécution se fait par la commande **Set-ExecutionPolicy** suivi du mode choisi.

Par exemple : `Set-ExecutionPolicy RemoteSigned` aura pour effet de placer la stratégie **RemoteSigned** comme stratégie d'exécution.

■ Astuce

Il n'est possible de modifier la stratégie d'exécution qu'avec les droits administrateur. Si vous travaillez avec un compte limité, choisissez de démarrer PowerShell avec un compte administrateur. Pour cela, utilisez la fonction « exécuter en tant que » disponible depuis un clic droit sur l'application.

Vous disposez également d'une deuxième commande **Get-ExecutionPolicy** qui permet, comme son nom l'indique de récupérer la stratégie d'exécution mise en place.

```
PS > Get-ExecutionPolicy
RemoteSigned
```

■ Astuce

Une solution consiste à modifier directement la clé de registre suivante :

HKEY_Local_Machine\SOFTWARE\Microsoft\PowerShell\1\ShellIds\Microsoft.PowerShell\ExecutionPolicy.

Microsoft livre un fichier .adm (à télécharger sur le site de Microsoft) pour configurer les stratégies d'exécution des machines par GPO (Group Policy Objects).

3.3 Les scripts provenant d'Internet

Si vous avez lu la partie précédente vous savez donc que les scripts créés localement ne sont pas soumis aux mêmes obligations que ceux provenant d'Internet.

Tout d'abord, avec la stratégie **RemoteSigned**, il ne vous sera pas possible d'exécuter des scripts téléchargés à partir d'Internet s'ils ne sont pas signés ou débloqués. Nous rajouterons même dans le cas de scripts signés, s'ils ne sont pas signés et provenant d'un éditeur approuvé.