CR431 – Scripts pour la Gestion de Système d'Exploitation avec Powershell

CERTIFICAT EN ANALYSE ET CYBERSÉCURITÉ OPÉRATIONNELLE

Devoir 1

Travail présenté à :

M. Philip Veilleux

Par :

Guillaume Carrier Couture - 2224664

Date de remise du travail :

Lundi 22 novembre 2023

Table des matières

Introduction

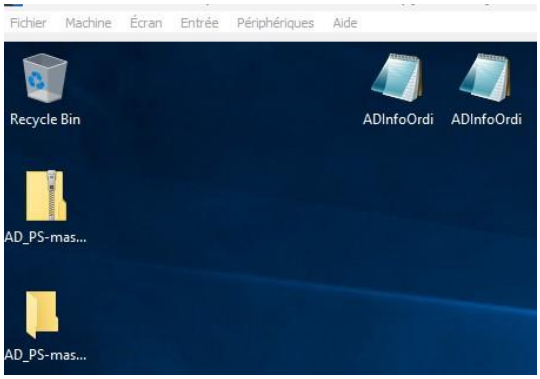
L'automatisation fait partie fondamentale du monde en constante évolution de la cybersécurité. En effet, cette pratique permet une meilleure gestion de temps en plus d'assurer la cohérence des opérations. Plus spécifiquement, Active Directory (AD), service d'annuaire de Microsoft, bénéficie grandement de l'automatisation des tâches. Powershell incarne une solution puissante à cet enjeu. Le concept de module, boîte à outils regroupant diverses membres PowerShell, telles des fonctions, dans l'optique d'un but commun, simplifie son utilisation en ajoutant des commandes pouvant être utilisées comme natives [1] [2].
Ce document consiste en la présentation du module « ADInfoOrdi ».
D'emblée, le cas d'utilisation sera présenté, suivi de l'installation du module, des dépendances du module, d'une explication pour un débutant en TI et d'une présentation des commandes.

1.0 Cas d'Utilisation

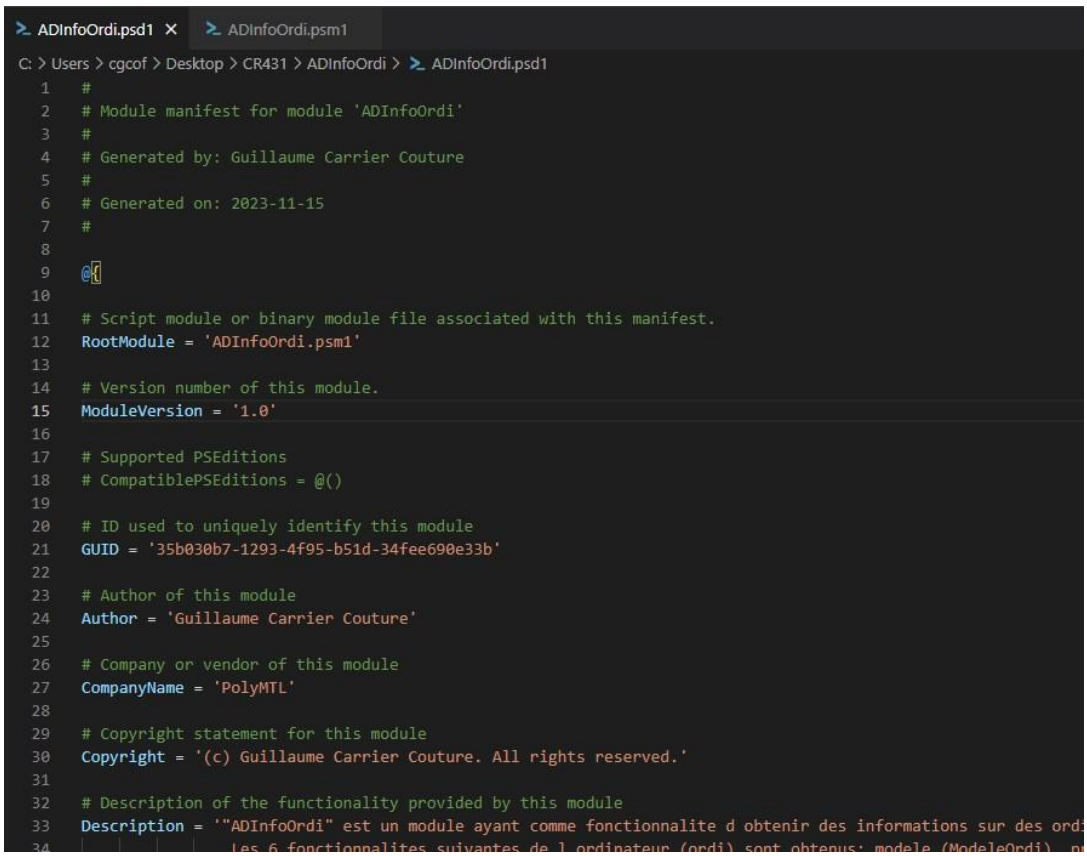
En tant que jeune étudiant à temps plein, je n'ai pas d'expérience professionnelle en TI à partir de laquelle je pourrais tirer un cas d'utilisation direct. Toutefois, j'aime créer puis configurer des machines virtuelles à l'aide de tutoriels GitHub ou YouTube pour mettre en place divers environnements et les tester à des fins d'apprentissage. Ces processus d'expérimentations m'ont permis, entre autres, d'explorer de nombreux systèmes d'exploitation et d'apprendre à effectuer des tests de pénétrations en utilisant Ubuntu, Kali et Commando VM.
Cette attitude m'a inspiré à créer le module PowerShell « ADInfoOrdi ». Le but du module est d'approfondir ma compréhension de PowerShell en testant, sur l'un de mes environnements AD, une extension ayant un cas d'utilisation précis.
« ADInfoOrdi » (Active Directory Information Ordinateur) est un module PowerShell ayant pour but d'obtenir des données spécifiques sur des ordinateurs. Le tout a été conçu dans l'optique du service d'annuaire Microsoft « Active Directory », mais peut être utilisé dans le cadre de n'importe quel environnement Windows. Comme précédemment mentionnée, l'automatisation est au cœur du module, cette fois-ci pour la gestion des informations sur l'équipement informatique.
Le module obtient les informations suivantes : le modèle, le processeur, la mémoire disponible, des données sur le disque (incluant le « deviceid », la taille en Go, l'espace libre en Go et de façon booléenne, si le lecteur a moins d'un quart d'espace disponible), l'adresse IP et l'heure du dernier démarrage. Toutes ces informations peuvent par la suite être obtenues de façon jointe grâce à la dernière fonction, « Get-InfoOrdi ». Le tout est expliqué plus en détail dans la section 4 du rapport.

2.0 Installation du Module

Procédons au processus d'installation du module « ADInfoOrdi ». D'emblée, j'ai importé les fichiers « ADInfoOrdi.psd1 » et « ADInfoOrdi.psm1 » sur mon contrôleur de domaine en les glissant simplement à partir de mon ordinateur local.

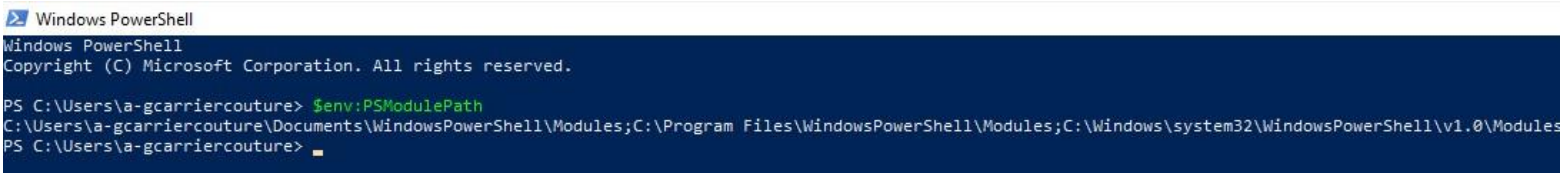


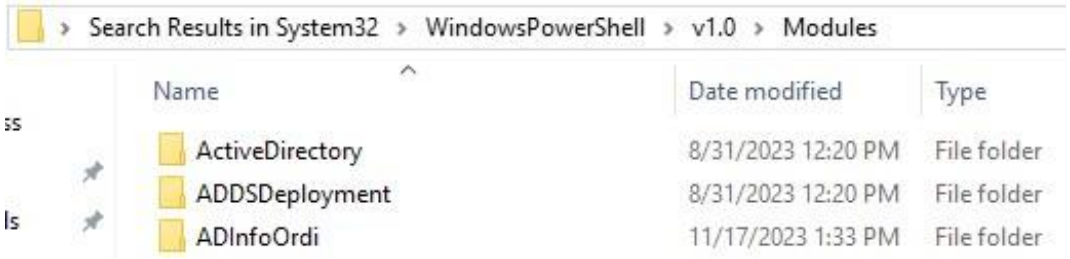
Deuxièmement, il faut s'assurer d'avoir créé un dossier conforme au nom du module (ADInfoOrdi) contenant ses fichiers « ADInfoOrdi.psm1 » et « ADInfoOrdi.psd1 », quoique le fichier « .psd1 » n'est pas strictement nécessaire au fonctionnement du module. Ce dernier est un manifeste de module PowerShell, soit un document contenant les métadonnées du module. Il contient aussi d'autres informations sur le module : son nom, son ID unique, le nom de l'auteur et plus [3]. D'ordre général, il est recommandé de simplement jeter un coup d'œil au fichier « .psd1 » avant d'utiliser un module afin d'obtenir plus d'informations sur celui-ci.



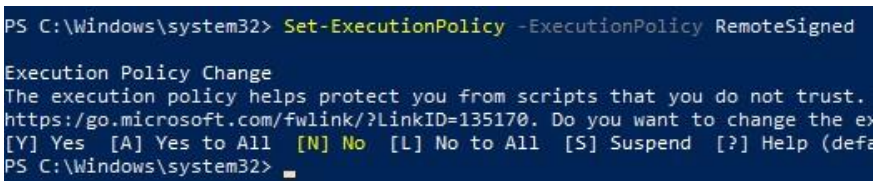
Continuons : le dossier doit être déposé dans le dossier « Modules » de PowerShell. Dans le cas de ma machine, le répertoire est trouvé dans le chemin suivant :
C:\Windows\system32\WindowsPowerShell\v1.0\Modules

Il est possible de trouver ce document grâce à la commande : $env:PSModulePath

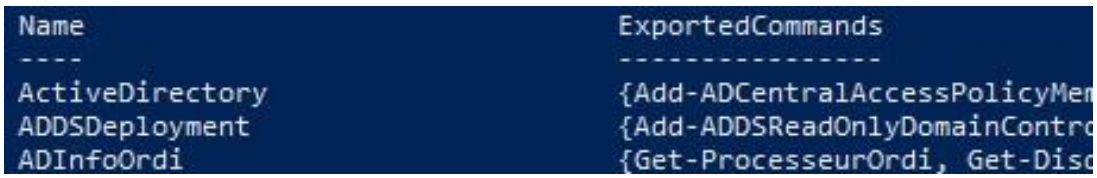




Il est par la suite impératif de définir la politique d'exécution PowerShell à « RemoteSigned » (signé à distance) pour que les scripts locaux puissent être exécutés. La commande est la suivante :
Set-ExecutionPolicy -ExecutionPolicy RemoteSigned



Ceci peut uniquement être fait en tant qu'administrateur. Il faut par la suite entrer « Y » pour confirmer le choix.
On peut finalement vérifier l'installation en obtenant la liste des modules installés :
Get-Module -ListAvailable

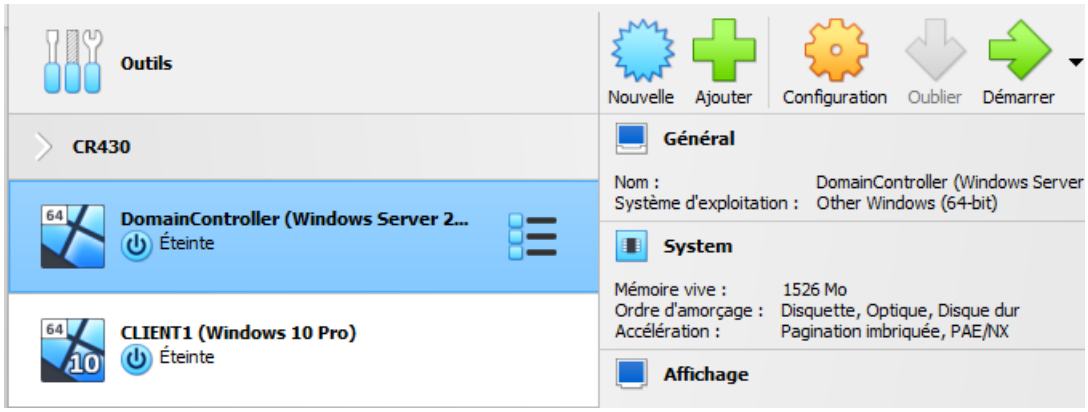


« ADInfoOrdi » est bel et bien présent.

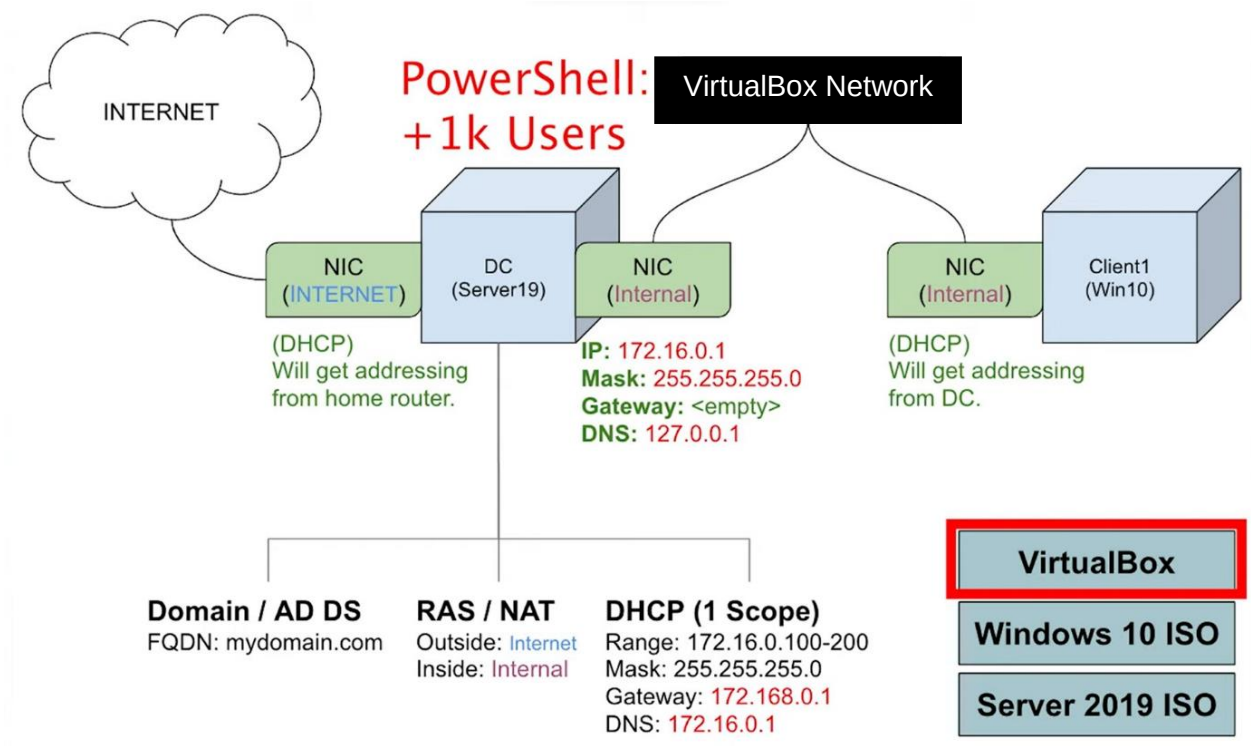
C'est tout, les fonctions du module peuvent désormais être utilisées comme natives.

3.0 Dépendances du Module

Ce module a été conçu pour fonctionner avec l'environnement d'annuaire Windows de base de Josh Makador « How to Setup a Basic Home Lab Running Active Directory (Oracle VirtualBox) | Add Users w/PowerShell » sur YouTube [4]. Cette dernière inclue une machine contrôleur de domaine Windows Server 2019 « DomainController » (Contrôleur de Domaine) et une machine cliente Windows 10 Pro « CLIENT1 ».



Voici, ci-dessous un schéma de l'environnement en question [3]. Plus d'informations techniques peuvent être obtenues dans la vidéo.



Toutefois, le module Powershell devrait fonctionner sur la plupart des contrôleurs de domaine Windows.

De plus, « ADInfoOrdi » a été conçu dans le cadre d'une seule scope DHCP. « Une étendue DHCP (Dynamic Host Configuration Protocol » est un regroupement administratif d'adresses IP pour les ordinateurs d'un sous-réseau que le serveur DHCP peut louer aux clients. » [5] Il est donc recommandé de ne pas l'utiliser dans d'autres contextes. Il est aussi recommandé d'utiliser PowerShell 5.1 (ou plus récent) et un environnement Windows 7 (ou plus récent). Faire autrement ne garantit pas le fonctionnement du module « ADInfoOrdi ».

4.0 Explication pour un Débutant en TI (Fonctionnement du Module)

Cette partie s'intéresse au code et à son fonctionnement.
Dans le document psm1, j'ai choisi de mettre les fonctions dans un ordre logique qui respecte un flux d'information croissant en termes de spécificité de l'information obtenue. Voici, ci-dessous, chacune d'entre elles.

La partie 4 du rapport a été conçue pour accompagner la lecture du document « ADOrdiInfo.psm1 ». Chaque fonction est commentée : les écrit fournis dans le code pourraient aussi davantage éclairer un débutant en TI.
Je tiens aussi à mentionner que le code PowerShell utilise la convention « PascalCase » [6]. Cette dernière décrète que les noms de l'ensemble des éléments, incluants noms de fonction et de variables, débutent par une majuscule, en plus de mettre une majuscule à tout changement de mot. Par exemple, « Get-InfoOrdi » y est conforme.

4.1 Get-ModeleOrdi :

La fonction « Get-ModeleOrdi » est la première du module et peut être trouvée à la ligne 29.

```
27
28  # Definition de la fonction pour obtenir le modele d'un ordinateur
29  function Get-ModeleOrdi {
30      <#
31      .SYNOPSIS
32      Obtient le modele d'un ordinateur (exemple: HP EliteDesk 800 G3 Desktop).
```

Au début de chacune d'entre elles, les fonctions PowerShell de « ADInfoOrdi » sont accompagnées d'une section de commentaire de l'aide (Help Comment-Based Help) [7]. Ce dernier contient un synopsis de la fonction, une description, les paramètres attendus et autres. Il est recommandé de se pencher sur cette section afin de mieux saisir son fonctionnement. Elles peuvent être obtenues directement dans le code ou dans la console PowerShell.

Pour le code, il suffit de s'aventurer dans le fichier :

```
<#
.SYNOPSIS
Obtient le modele d'un ordinateur (exemple: HP EliteDesk 800 G3 Desktop).

.DESCRIPTION
Obtient le modele d'un ordinateur ou d'un groupe d'ordinateurs.

.PARAMETER Name
Le modele de l'ordinateur avec son nom sera retourne. Par defaut, il s'agit de l'ordinateur local.

.INPUTS
Prend un tableau de noms d'ordinateurs ou d'objets AD en entree via le pipeline.

.OUTPUTS
Renvoie des objets PS pour les ordinateurs qui sont passes, incluant le nom de l'ordinateur et le modele.

.EXAMPLE
Get-ModeleOrdi -Name 'ordi1'

Renvoie un objet PS avec le nom de l'ordinateur et le modele de 'ordi1'.

.EXAMPLE
'ordi1','ordi2' | Get-ModeleOrdi

Renvoie un objet PS pour chaque ordinateur contenant le nom de l'ordinateur et le modele.

.EXAMPLE
Get-OrdinateurOU -UniteOrganisationnelle 'ServiceA' | Get-ModeleOrdi

Renvoie un objet PS pour chaque ordinateur dans l'unite organisationnelle 'ServiceA' Active Directory contenant le nom de l'ordinateur et le modele.
#>
```

Dans la console PowerShell, j'ai appelé comme exemple le synopsis de « Get-ModeleOrdi » grâce à la commande : (Get-Help Get-ModeleOrdi -Full).Synopsis

```
Administrator: Windows PowerShell
PS C:\Windows\system32> (Get-Help Get-ModeleOrdi -Full).Synopsis
Obtient le modele d'un ordinateur (exemple: HP EliteDesk 800 G3 Desktop).
PS C:\Windows\system32>
```

Elles contiennent aussi toutes une section « Param() », comportant la déclaration des paramètres dynamiques (qui peuvent être modifiés dans l'appel de la fonction) [8].

```
[CmdletBinding()]
Param(
    [parameter(ValueFromPipeline=$True,ValueFromPipelineByPropertyName=$True)]
    [Alias('NomOrdinateur')]
    [string]$Nom = $env:COMPUTERNAME
)
```

La fonction « Get-ModeleOrdi » a pour but, comme le révèle son titre, d'obtenir le modèle d'un ordinateur ou d'un groupe d'ordinateur. La fonction fait appel à une commande PowerShell (native) nommée « Get-CimInstance » pour interroger l'ordinateur et obtenir des détails matériels spécifiques, dans ce cas-ci, le modèle. Après avoir obtenu la donnée, elle organise le tout pour présenter clairement le nom de l'ordinateur et son modèle associé.

4.2 Get-ProcesseurOrdi :

La fonction « Get-ProcesseurOrdi » est la deuxième du module et peut être trouvée à la ligne 85.

```
84  # Definition de la fonction pour obtenir le processeur d'un ordinateur
85  function Get-ProcesseurOrdi {
86      <#
87      .SYNOPSIS
88      Obtient le processeur d'un ordinateur(ex: Intel Core i7-8700K).
```

Cette fonction effectue une tâche semblable à celle présentée à la section 4.1, cette fois, en obtenant le processeur de l'ordinateur ou du groupe d'ordinateur. Elle fait aussi usage de « Get-CimInstance ». Les informations obtenues sont ajoutées à la liste « $ProcesseursOrdi », renvoyée dans le cadre de la section « end ».

4.3 Get-MemoireOrdi :

La fonction « Get-MemoireOrdi » est la troisième du module et peut être trouvée à la ligne 141.

```
139
140 # Definition de la fonction pour obtenir la memoire en Go d'un ordinateur
141 function Get-MemoireOrdi {
142     <#
143     .SYNOPSIS
144     Obtient la memoire en Go d'un ordinateur (ex: 60 Go).
```

Elle permet d'obtenir la mémoire en Go d'un ordinateur ou d'un groupe d'ordinateurs. Si elle n'est pas en ligne et « Test-Connection » ne porte pas fruit, « $Nom est hors ligne. » sera renvoyé. Elle utilise encore une fois « Get-CimInstance ». Elle convertit par la suite la valeur « TotalPhysicalMemory » en Go, le stocke dans l'objet « PSCustomObject » puis envoie le nom de l'ordinateur accompagné de sa mémoire.

4.4 Get-DisqueDurOrdi :

La fonction « Get-DisqueDurOrdi » est la quatrième du module et peut être trouvée à la ligne 204.

```
203
204 # Definition de la fonction pour obtenir des informations sur les disques d'un ordinateur
205 function Get-DisqueDurOrdi {
206     <#
207     .SYNOPSIS
208     Obtient des informations sur les disques d'un ordinateur.
```

Cette dernière utilise la même méthode que plus haut, soit en utilisant « Get-CimInstance ». Toutefois, plus d'informations sont obtenues dans la liste : le « Device ID » du disque, la taille en Go, l'espace libre en Go et une valeur en booléenne (vrai ou faux) de si le disque à moins de 25% d'espace libre.

```
process{
    $ListeInformationsDisque += Get-CimInstance -ComputerName $Nom -ClassName win32_logicaldisk
        Where-Object -Property DriveType -EQ 3 |
        Select-Object -Property @{n="Ordinateur";e={$Nom}},`
        @{n="DeviceID";e={$_.deviceid}},`
        @{n="NomVolume";e={$_.volumename}},`
        @{n="TailleGo";e={$_.size / 1GB -as [int]}},`
        @{n="EspaceLibreGo";e={$_.freespace / 1GB -as [int]}},`
        @{n="MoinsDe25Pourcent";e={if(($_.freespace / $_.size) -le 0.25){"Vrai"}else{"Faux"}}}
```

4.5 Get-AdresseIPOrdi :

La fonction « Get-AdresseIPOrdi » est la cinquième du module et peut être trouvée à la ligne 273.

```
271
272   # Definition de la fonction pour obtenir l'adresse IPv4 d'un ordinateur
273   function Get-AdresseIPOrdi {
274       <#
275       .SYNOPSIS
276       Obtient l'adresse IPv4 d'un ordinateur.
```

La fonction récupère l'adresse IPV4 d'un ordinateur. En utilisant la commande `Resolve-DnsName`, elle crée un objet PowerShell contenant le nom de l'ordinateur et son adresse IP, renvoyant finalement cette liste d'objets.

4.6 Get-DernierDemarrageOrdi :

La fonction « Get-DernierDemarrageOrdi » est la sixième du module et peut être trouvée à la ligne 321.

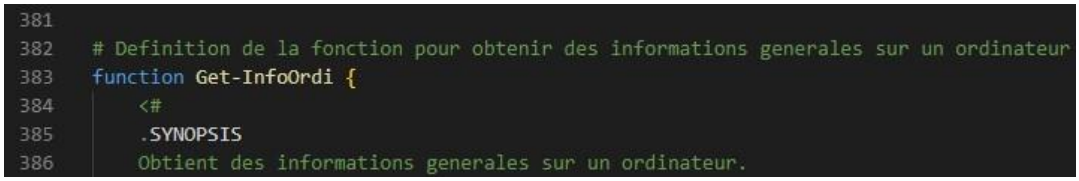
```
319
320   # Definition de la fonction pour obtenir la derniere heure de demarrage d'un ordinateur
321   function Get-DernierDemarrageOrdi {
322       <#
323       .SYNOPSIS
324       Obtient la derniere heure a laquelle un ordinateur a demarre.
```

« Get-DernierDemarrageOrdi » sort la dernière heure de démarrage de l'ordinateur en question. D'emblée, « Get-CimInstance » avec la classe « Win32_OperatingSystem » obtient

la dernière heure de démarrage puis le résultat est formaté. La liste résultante est finalement renvoyée.

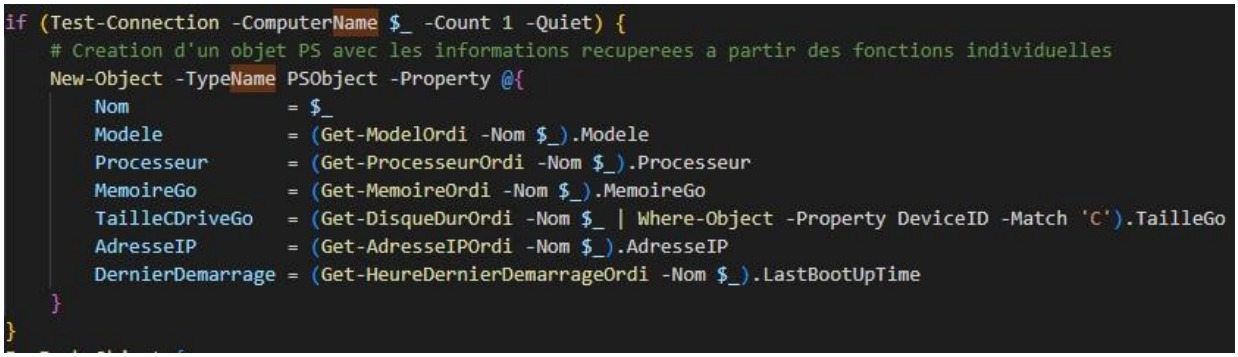
4.7 Get-InfoOrdi :

La fonction « Get-InfoOrdi » est la septième du module et peut être trouvée à la ligne 383.



```
381
382  # Definition de la fonction pour obtenir des informations generales sur un ordinateur
383  function Get-InfoOrdi {
384      <#
385      .SYNOPSIS
386      Obtient des informations generales sur un ordinateur.
```

Cette fonction regroupe les fonctions précédemment mentionnées afin d'obtenir un ensemble d'informations sur un ordinateur ou un groupe d'ordinateur. D'emblée, deux listes sont créées dans la section « Begin » (une pour le nom de l'ordi et l'autre pour stocker les informations obtenues. Dans le processus, les fonctions « Get-ModelOrdi », « Get-ProcesseurOrdi », « Get-MemoireOrdi », « Get-DisqueDurOrdi », « Get-AdresseIPOrdi » puis « Get-HeureDernierDemarrageOrdi » sont toutes mises en marche. Finalement, la liste « $ListeInfoOrdinateur » est retournée, incluant les informations obtenues.



```
if (Test-Connection -ComputerName $_ -Count 1 -Quiet) {
    # Creation d'un objet PS avec les informations recuperees a partir des fonctions individuelles
    New-Object -TypeName PSObject -Property @{
        Nom             = $_
        Modele          = (Get-ModelOrdi -Nom $_).Modele
        Processeur      = (Get-ProcesseurOrdi -Nom $_).Processeur
        MemoireGo       = (Get-MemoireOrdi -Nom $_).MemoireGo
        TailleCDriveGo  = (Get-DisqueDurOrdi -Nom $_ | Where-Object -Property DeviceID -Match 'C').TailleGo
        AdresseIP       = (Get-AdresseIPOrdi -Nom $_).AdresseIP
        DernierDemarrage = (Get-HeureDernierDemarrageOrdi -Nom $_).LastBootUpTime
    }
}
```

5.0 Commandes

Voici, ci-dessous, l'utilisation pratique de chaque fonction contenue dans le module. Chacune d'entre elles est accompagnée d'un exemple. Ci-dessous, dans la section « Notes », sont aussi montrées d'autres fonctionnalités pouvant s'appliquer aux fonctions, en plus des paramètres utilisés.

Notes :
- Il est possible de renvoyer plusieurs ordinateurs en faisant usage du pipeline (exemple : 'ordi1','ordi2' | Get-ModeleOrdi).

- Il est aussi possible d'obtenir les informations sur les disques pour tous les ordinateurs dans AD (exemple : Get-ADComputer -Filter * | Get-DisqueDurOrdi). Le paramètre « Filter » suivi de « * » implique que tous les objets (donc les ordinateurs d'AD par « ADComputer ») seront récupérés. Ce n'est pas lié directement au module du rapport mais tout de même utile.

-De plus, si le paramètre « -Nom » n'est pas spécifié, les informations de l'ordinateur local seront retournées. Ce dernier spécifie le nom de l'ordinateur qui est examiné.

5.1 Get-ModeleOrdi :

Get-ModeleOrdi -Nom 'DOMAINCONTROLLE'

```
PS C:\Windows\system32> Get-ModeleOrdi -Nom 'DOMAINCONTROLLE'

Nom             Modele
---             ------
DOMAINCONTROLLE VirtualBox
```

5.2 Get-ProcesseurOrdi :

Get-ProcesseurOrdi -Nom 'DOMAINCONTROLLE'

```
PS C:\Windows\system32> Get-ProcesseurOrdi -Nom 'DOMAINCONTROLLE'

Nom             Processeur
---             ----------
DOMAINCONTROLLE Intel(R) Core(TM) i5-1035G1 CPU @ 1.00GHz
```

5.3 Get-MemoireOrdi :

Get-MemoireOrdi -Nom 'DOMAINCONTROLLE'

```
PS C:\Windows\system32> Get-MemoireOrdi -Nom 'DOMAINCONTROLLE'

Nom             MemoireGo
---             ---------
DOMAINCONTROLLE       1.5
```

5.4 Get-DisqueDurOrdi :

Get-DisqueDurOrdi -Nom 'DOMAINCONTROLLE'

```
PS C:\Windows\system32> Get-DisqueDurOrdi -Nom 'DOMAINCONTROLLE'


Ordinateur         : DOMAINCONTROLLE
DeviceID           : C:
TailleGo           : 19
EspaceLibreGo      : 6
MoinsDe25Pourcent  : Faux
```

5.5 Get-AdresseIPOrdi :

Get-AdresseIPOrdi -Nom 'DOMAINCONTROLLE'

```
PS C:\Windows\system32> Get-AdresseIPOrdi -Nom 'DOMAINCONTROLLE'

Nom             AdresseIP
---             ---------
DOMAINCONTROLLE 172.16.0.1
```

5.6 Get-DernierDemarrageOrdi :

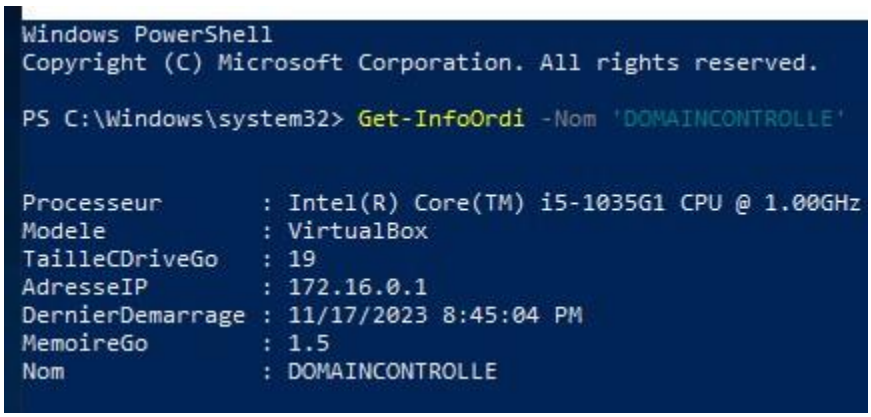
Get-DernierDemarrageOrdi -Nom 'DOMAINCONTROLLE'

```
PS C:\Windows\system32> GEt-DernierDemarrageOrdi -Nom 'DOMAINCONTROLLE'

Nom             LastBootUpTime
---             --------------
DOMAINCONTROLLE 11/17/2023 8:45:04 PM
```

5.7 Get-InfoOrdi :

Get-InfoOrdi -Nom 'DOMAINCONTROLLE'



Conclusion

Au début du texte, j'ai exprimé mon initiative d'approfondir mes connaissances en PowerShell en faisant l'usage d'un module construit dans le but d'obtenir les informations des ordinateurs présents dans mon environnement d'apprentissage contrôlé. J'ai par la suite expliqué chaque fonction présente dans le code, puis fournis un exemple, accompagné des notes sur les fonctionnalités, par fonction. En bref, l'automatisation fait partie fondamentale de la gestion d'AD. Dans le cadre de ce travail, « ADInfoOrdi » développe justement ce concept : à l'aide de nombreuses fonctions, il rend facile l'obtention des informations d'ordinateurs d'un setup Active Directory.

Bibliographie

[1] sdwheeler, "about Modules - PowerShell," *learn.microsoft.com*. https://learn.microsoft.com/en-us/powershell/module/microsoft.powershell.core/about/about_modules?view=powershell-7.3 (accédé le 16 novembre 2023).

[2] Polytechnique Montréal. (2023), Philip Veilleux, CR431E Gestion des Risques de l'Information (Cours 2) : PowerShell : Éléments de Base [En ligne]

[3] sdwheeler, "How to Write a PowerShell Module Manifest - PowerShell," *learn.microsoft.com*, 6 mars 2023. https://learn.microsoft.com/en-us/powershell/scripting/developer/module/how-to-write-a-powershell-module-manifest?view=powershell-7.3 (accédé le 17 novembre 2023).

[4] "How to Setup a Basic Home Lab Running Active Directory (Oracle VirtualBox) | Add Users w/PowerShell," *www.youtube.com*, 4 janvier 2021 https://www.youtube.com/watch?v=MHsI8hJmggI&t=11s (accédé le 17 novembre 2023).

[5] robinharwood, "Étendues DHCP dans Windows Server," *learn.microsoft.com*, 7 octobre 2023. https://learn.microsoft.com/fr-fr/windows-server/networking/technologies/dhcp/dhcp-scopes (accédé 17 novembre 2023).

[6] "Code Layout and Formatting", *PowerShell Practice and Style*, https://poshcode.gitbook.io/powershell-practice-and-style/style-guide/code-layout-and-formatting (accédé le 17 Novembre 2023).

[7] sdwheeler, "about Comment Based Help - PowerShell," *learn.microsoft.com*, 19 septembre 2022. https://learn.microsoft.com/en-us/powershell/module/microsoft.powershell.core/about/about_comment_based_help?view=powershell-7.3 (accédé le 16 novembre 2023).

[8] sdwheeler, "about Functions Advanced Parameters - PowerShell," *learn.microsoft.com*, Jun. 23, 2023. https://learn.microsoft.com/en-us/powershell/module/microsoft.powershell.core/about/about_functions_advanced_parameters?view=powershell-7.3 (accédé le 16 novembre 2023).