Travaux Pratiques - Projet Sécurité & Débogage

Objectifs :

Ce projet a pour but de vous familiariser avec la gestion de la sécurité sur Windows via PowerShell, la création et le débogage de scripts, ainsi que la gestion des erreurs. Vous utiliserez Visual Studio Code et Git pour structurer et versionner vos travaux.

Prérequis :

- Visual Studio Code installé

- Git installé

- PowerShell (version 5.1 minimum recommandée)

- Système Windows (Home ou Pro)

Organisation du projet :

1. Créer un dossier de travail sur votre machine.

2. Initialiser un dépôt Git local dans ce dossier.

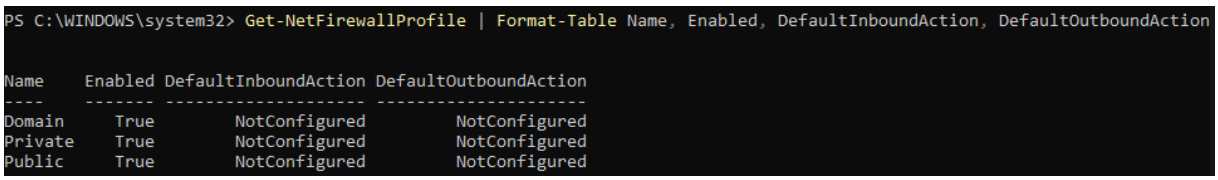
3. Créer plusieurs scripts PowerShell correspondant aux exercices ci-dessous.

4. Documenter vos scripts avec des commentaires expliquant chaque étape.

5. Valider régulièrement vos modifications avec Git.

1. Gestion des règles de pare-feu Windows :

- Affichez la configuration actuelle des profils de pare-feu.



- Ajoutez une règle de pare-feu qui bloque le trafic sortant sur un port TCP précis (au choix).

```
PS C:\WINDOWS\system32>
>> New-NetFirewallRule -DisplayName Blocage_Traffic_Sortant_8080 `
>>     -Direction Outbound `
>>     -Action Block `
>>     -Protocol TCP `
>>     -LocalPort 8080 `
>>     -Profile Any


Name                          : {237e4356-912d-4a3d-a237-9aa42f60ff2a}
DisplayName                   : Blocage_Traffic_Sortant_8080
Description                   :
DisplayGroup                  :
Group                         :
Enabled                       : True
Profile                       : Any
Platform                      : {}
Direction                     : Outbound
Action                        : Block
EdgeTraversalPolicy           : Block
LooseSourceMapping            : False
LocalOnlyMapping              : False
Owner                         :
PrimaryStatus                 : OK
Status                        : La règle a été analysée à partir de la banque. (65536)
EnforcementStatus             : NotApplicable
PolicyStoreSource             : PersistentStore
PolicyStoreSourceType         : Local
RemoteDynamicKeywordAddresses : {}
PolicyAppId                   :
PackageFamilyName             :
```

- Vérifiez que la règle est bien appliquée.

```
PS C:\WINDOWS\system32> Get-NetFirewallRule -DisplayName Blocage_Traffic_Sortant_8080


Name                          : {237e4356-912d-4a3d-a237-9aa42f60ff2a}
DisplayName                   : Blocage_Traffic_Sortant_8080
Description                   :
DisplayGroup                  :
Group                         :
Enabled                       : True
Profile                       : Any
Platform                      : {}
Direction                     : Outbound
Action                        : Block
EdgeTraversalPolicy           : Block
LooseSourceMapping            : False
LocalOnlyMapping              : False
Owner                         :
PrimaryStatus                 : OK
Status                        : La règle a été analysée à partir de la banque. (65536)
EnforcementStatus             : NotApplicable
PolicyStoreSource             : PersistentStore
PolicyStoreSourceType         : Local
RemoteDynamicKeywordAddresses : {}
PolicyAppId                   :
PackageFamilyName             :
```

- Supprimez la règle créée.

```
PS C:\WINDOWS\system32> Remove-NetFirewallRule -DisplayName Blocage_Traffic_Sortant_8080
```

Objectif : Comprendre comment gérer le pare-feu avec PowerShell et manipuler des règles.

2. Création et débogage d'un script PowerShell

- Créez un script PowerShell prenant un paramètre en entrée (par exemple un nom).

```
param(
    [string]$Nom = "Utilisateur"
)
```

- Affichez des informations liées à l'exécution du script, notamment les arguments reçus et le chemin du script.

```
PS C:\Users\Valentin Foure\Desktop\Dossier\Windows PowerShell\Projet\PowerShell\TP_SecuriteDebogage> .\Debug.ps1
Nom fourni : Utilisateur
Chemin du script : C:\Users\Valentin Foure\Desktop\Dossier\Windows PowerShell\Projet\PowerShell\TP_SecuriteDebogage\Debug.ps1
Dossier du script : C:\Users\Valentin Foure\Desktop\Dossier\Windows PowerShell\Projet\PowerShell\TP_SecuriteDebogage
```

- Mettez en place un point d'arrêt dans le script.

```
# Point d'arrêt pour le débogage
break
```

- Exécutez le script et utilisez les outils de débogage PowerShell pour analyser le comportement.

```
1   param(
2       [string]$Nom = "Utilisateur"
3   )
4
5   Write-Output "Nom fourni : $Nom"
6   Write-Output "Chemin du script : $PSCommandPath"
7   Write-Output "Dossier du script : $(Split-Path -Path $PSCommandPath)"
8
9   # Point d'arrêt pour le débogage
10  break
11
12  Write-Output "Script terminé"
```

Objectif : Apprendre à créer des scripts avec paramètres et utiliser les fonctions de débogage intégrées.

3. Gestion des erreurs dans un script

- Écrivez un script qui tente d'accéder à un fichier qui n'existe pas.

```
$filePath = "C:\fichier_inexistant.txt"
```

- Implémentez un bloc try/catch/finally pour capturer et gérer l'erreur.

```
try {
    Write-Output "Tentative d'accès au fichier : $filePath"
    $contenu = Get-Content -Path $filePath -ErrorAction Stop
    Write-Output $contenu
}
catch {
    Write-Error "Erreur : Impossible d'accéder au fichier. $_"
}
finally {
    Write-Output "Bloc finally exécuté. Nettoyage terminé."
}
```

- Affichez un message utilisateur clair lorsque l'erreur est détectée.

- Assurez-vous que le script continue ou se termine proprement.

```
Write-Output "Fin du script."
```

Objectif : Savoir gérer les exceptions et assurer la robustesse d'un script.

Travail à rendre :

- Tous vos scripts doivent être déposés dans le dossier du projet.

- Chaque script doit comporter des commentaires expliquant son fonctionnement.

- Rédigez un fichier README.md qui décrit votre démarche, les choix techniques, et les difficultés rencontrées.