# ДОДАТОК А

## Варіант написання скрипта для аудиту комп'ютерів в мережі з установленими ролями Active Directory Domain Service

На рис. А.1 зображений скрипт LaunchUSBHiddenNetworks.

```
$ Salida = invoke-command -ComputerName (Get-Content servers.txt)
-FilePath 'PathToScript \ RecollectUSBData.ps1'
-Credential testdomain \ administrador
```

Рисунок А.1 – Скрипт LaunchUSBHiddenNetworks

Робота цього скрипта заснована на команді "Invoke–Command". Скрипт дозволяє з'єднатися з комп'ютером в мережі, якщо в якості параметрів передати ім'я або IP-адресу і шлях до ще одного скрипту для подальшого запуску в PowerShell.

У параметрі –ComputerName задається ім'я комп'ютера (або списку комп'ютерів) для аудиту, які знаходяться всередині Active Directory. Список імен можна перерахувати через кому або, як це було зроблено в скрипті, передати у вигляді текстового файлу (servers.txt).

У параметрі –FilePath передається ім'я скрипта, який буде збирати дані на комп'ютерах. У параметрі –Credential передається обліковий запис адміністратора домену Щоб дозволити початок на віддаленому комп'ютері (домен – testdomain, користувач – administrator).

Результати роботи скрипта записуються в об'єкт $ salida. Отримана інформація буде зберігатися в файлі CSV з ім'ям "USBDATA.csv". На рис.А.2 зображений результат роботи скрипта. В файл CSV було виведено ім'я комп'ютера, IP (в форматі IPv4), ім'я USB-пристрою, ID (унікальний ідентифікатор).

```
PC001,192.168.1.16,Kingston DataTraveler G3 USB Device,{2057d6e6-7725-52d5-8d5e-3fdab3357470}
PC001,192.168.1.16,SanDisk Cruzer Blade USB Device,{1df90487-d45c-5a58-8509-dff4fae7bca6}
PC002,192.168.1.15,Kingston DataTraveler G3 USB Device,{2057d6e6-7725-52d5-8d5e-3fdab3357470}
PC002,192.168.1.15,SanDisk Cruzer Blade USB Device,{1df90487-d45c-5a58-8509-dff4fae7bca6}
```

Рисунок А.2 – Результат роботи скрипта LaunchUSBHiddenNetworks

# ДОДАТОК Б

## Варіант скрипта для сбору інформації стосовно USB-пристроїв на ком'ютерах в мережі з установленими ролями Active Directory Domain Service

Цей скрипт збирає всю інформацію, що стосується USB-пристроїв, що підключаються до комп'ютера і запускається локально. Інформація збирається з певною гілки реєстру ОС Windows. Розглянемо всі рядки скрипту по черзі.

```
$ USBDevices = @ ()
$ USBContainerID = @ ()
$ USBComputerName = @ ()
$ USBComputerIP = @ ()
$ SubKeys2 = @ ()
$ USBSTORSubKeys1 = @ ()
$ Hive = "LocalMachine"
$ Key = "SYSTEM \ CurrentControlSet \ Enum \ USBSTOR"
```

Змінні $ Hive і $ Key зберігають повний шлях до гілки реєстру, де знаходиться інформації щодо підключаються USB-пристроїв. Значення "LocalMachine" еквівалентно розділу HKLM або HKEY_LOCAL_MACHINE.

```
$ ComputerName = $ Env: COMPUTERNAME
$ ComputerIP = $ localIpAddress = ((ipconfig | findstr [0–9]. \.) [0]). Split () [– 1]
```

Ім'я та IP-адреса локального комп'ютера зберігаються в змінних $ ComputerName і $ ComputerIP відповідно.

```
$ Reg = [Microsoft.Win32.RegistryKey] :: OpenRemoteBaseKey ($ Hive, $ Computer)
$ USBSTORKey = $ Reg.OpenSubKey ($ Key)
$ Nop = $ false
```

В об'єкті $ Reg зберігається запит до реєстру на базі команди OpenRemoteBaseKey, в яку передаються параметри $ Hive і $ Computer. У підсумку буде отримана гілка, в котрій буде здійснюватися пошук. Змінна $ nop використовується далі для управління потоком виконання.

```
Try {
$ USBSTORSubKeys1 = $ USBSTORKey.GetSubKeyNames ()
```

```
}
Catch
{
Write–Host "Computer:", $ ComputerName –foregroundcolor "white" –
backgroundcolor "red"
Write–Host "No USB data found"
$ Nop = $ true
}
```

Блок Try – Catch відповідає за обробку помилок в разі, якщо інформації щодо USB-пристроїв знайдено не буде. Якщо ніяких даних не знайдено, в змінній $ nop присвоюється значення змінної $ true для того, щоб уникнути виконання процесу ідентифікації та отримання інформації про USB-пристроях.

```
if (–Not $ nop)
```

У разі, якщо що–небудь знайдено, пов'язане з тим, що підключається USB-пристроєм, будуть запущені наступні блоки:

Блок 1:

```
ForEach ($ SubKey1 in $ USBSTORSubKeys1)
{
$ Key2 = "SYSTEM \ CurrentControlSet \ Enum \ USBSTOR \ $ SubKey1"
$ RegSubKey2 = $ Reg.OpenSubKey ($ Key2)
$ SubkeyName2 = $ RegSubKey2.GetSubKeyNames ()
$ Subkeys2 + = "$ Key2 \ $ SubKeyName2"
$ RegSubKey2.Close ()
}
```

Кожен запис в гілці реєстру, де відбувається пошук, пов'язаний з окремим USB-пристроєм. Кожен елемент зберігається в матриці @ Subkeys2.

Блок 2:

```
ForEach ($ Subkey2 in $ Subkeys2)
{
$ USBKey = $ Reg.OpenSubKey ($ Subkey2)
$ USBDevice = $ USBKey.GetValue ( 'FriendlyName')
$ USBContainerID = $ USBKey.GetValue ( 'ContainerID')
If ($ USBDevice)
{
```

```
$ USBDevices + = New–Object –TypeName PSObject –Property @ {
USBDevice = $ USBDevice
USBContainerID = $ USBContainerID
USBComputerName = $ ComputerName
ComputerIP = $ ComputerIP
}
}
$ USBKey.Close ()
}
```

Цей блок аналізує інформацію по кожному USB-пристрою, отриману в Блоці 1, яка зберігається в матриці @ Subkeys2. Для кожного елемента, що має значення в поле $ USBDevice, витягується ідентифікатор USB-пристрої (змінна USBContainerID). Ім'я та IP-адреса комп'ютера також зберігаються у відповідних змінних, щоб згодом ця інформація була додана в CSV-файл.

Блок 3:

```
for ($ i = 0; $ i –lt $ USBDevices.length; $ i ++) {
$ IDUnico = $ USBDevices [$ i] | Select –ExpandProperty "USBContainerID"
$ USBNombre = $ USBDevices [$ i] | Select –ExpandProperty "USBDevice"
Write–Host "Computer:", $ ComputerName –foregroundcolor "black" –backgroundcolor "green"
Write–Host "IP:", $ ComputerIP
Write–Host "USB found:", $ USBNombre
Write–Host "USB ID:", $ IDUnico
Echo "$ ComputerName, $ ComputerIP, $ USBNombre, $ IDUnico"
}
```

Третій блок відображає інформацію, зібрану з віддаленого комп'ютера. Команда Write-Host використовується для виведення інформації на екран. Команда Echo – для виведення даних, які згодом будуть записані в файл CSV.