Praca w systemie za pomocą konsoli. Uruchamianie programów z uprawnieniami administratora.

System Windows korzysta głównie z interfejsu graficznego. Interfejs ten to jednak najczęściej jedynie program, uruchamiany jako składnik systemu. W podstawie system uruchamia warstwę graficzną, pozwalającą na uruchamianie innych programów (odpowiedni system bibliotek). Następnie uruchamiany jest program logonUI.exe, który odpowiada za graficzny ekran logowania. Po wpisaniu hasła możliwe staje się uruchomienie kolejnego programu (w tym wypadku explorer.exe), który to towarzyszy nam przez cały czas działania systemu Windows.

System Windows to jednak nie tylko okienka. To głównie zestaw programów i narzędzi, które działają w konsoli i które można bez większych problemów wywołać poprzez wpisanie ich nazwy/ nazwy odpowiedniego polecenia systemowego. Wbrew pozorom obsługa systemu z linii poleceń (w tym w najnowszej wersji konsoli PowerShell) jest znacznie szybsza i przyjemniejsza niż wyszukiwanie kolejnych opcji w kolejnych warstwach okien.

INFORMACJA: W poniższym materiale wykorzystywane będą polecenia PowerShell. Różne wersje systemu Windows posiadają różne wersje tegoż interpretera. Aby mieć pewność, że wszystko zadziała tak jak w przykładach należy zaktualizować swój system do wersji co najmniej 5.1. Pobrać go można z tego odnośnika:
https://www.microsoft.com/en-us/download/details.aspx?id=54616
oraz
https://www.microsoft.com/en-us/download/details.aspx?id=50395

1. Uruchamianie zadań ze zmienionymi uprawnieniami.

Od czasów Windows Vista w systemie nie mamy bezpośredniego dostępu do plików, folderów i aplikacji wymagających od nas podwyższonych uprawnień. Możemy je pozyskać tylko w przypadku gdy:
- posiadamy dostęp do konta Administrator i jest ono odblokowane
- należymy do grupy wbudowanej Administratorzy i/lub znamy hasło dla użytkownika z tejże grupy

W innych przypadkach nie wykonamy operacji z ikoną tarczy (wymagających dodatkowych uprawnień). Docelowego użytkownika rozwiązanie to chroni chociażby przed instalacją w systemie ukrytego oprogramowania, inwazyjną (lecz cichą) zmianą ustawień systemowych czy też podmianą plików systemowych na fałszywe (szkodliwe).

W interfejsie graficznym możemy uruchamiać wybrane przez nas aplikacje np. jako administrator (klikając na odpowiednią opcję w menu, prawym menu bądź zostaniemy poproszeni o autoryzację w chwili wykonywania tego rodzaju operacji. Co jednak, jeżeli chcemy tego typu operację wykonać z linii poleceń/uruchomić aplikację z uprawnieniami innego użytkownika lecz niekoniecznie administratora?

W tym wypadku z pomocą przychodzi linia poleceń. W linii poleceń (klasycznej) mamy polecenie o nazwie:

runas

Polecenie to pozwala na uruchamianie aplikacji jako dowolny inny użytkownik dostępny w systemie. Dodatkowym atutem jest możliwość uruchomienia bezprofilowego (np. bez zmiennych lokalnych) oraz zachowanie poświadczeń (kolejne uruchomienie wskazanego programu z konta wywołującego nie będzie poprzedzone pytanie o hasło).

Przykład wywołania:

runas /user:staszek ‘C:\Program Files\Oracle\VirtualBox.exe’

uruchomi to program VirtualBox, który będzie korzystał z folderu domowego (oraz uprawnień) użytkownika staszek, nie modyfikując jednocześnie profilu użytkownika wywołującego polecenie.

PowerShell przynosi kolejne polecenia do wywołania aplikacji (wraz z odpowiednimi uprawnieniami):

Start-Process
Invoke-Command
Invoke-WmiMethod

Ostatnie z wymienionych pozwala na wykonywanie zapytań do obiektu WMI (Windows Management Instrumentation). Dzięki temu możemy wywoływać polecenia operujące np. na interfejsach sieciowych, procesach systemowych itp. Polecenie nie nadaje się natomiast do działania na standardowych poleceniach.

Drugie polecenie, Invoke-Command, pozwala na wywoływanie zapytań zdalnie. Oznacza to w praktyce, że możemy wydać serię poleceń, takich jak uruchamianie aplikacji, włączanie czy wyłączanie usług systemowych, blokowanie komputera itp. na maszynach, do których nie mamy fizycznie dostępu. Wymagany jest natomiast aktywny odpowiedni moduł systemu WinRM (Windows Remote Management), który pozwoli na autoryzację na danym systemie. UWAGA – polcenie może być wykonane na urządzeniu lokalnym, musi ono jednak mieć aktywny WinRM!

Ostatnie polecenie, Start-Process, jest połączeniem polecenia start i runas. Pozwala na uruchamianie pojedynczego polecenia – zarówno konsolowego, jak i graficznego. Domyślnie wszystkie uruchamiane przez to polecenie aplikacje będą miały uprawnienia osoby wywołującej. Jednak uruchomienie z parametrem Credential można wskazać konkretnego użytkownika, którego uprawnienia mają być zastosowane do aplikacji.. Przykładowo:

Start-Process -Credential staszek -FilePath ‘C:\Program Files\Oracle\VirtualBox.exe’

uruchomi VirtualBox z uprawnieniami użytkownika staszek. Dodatkowo wywołując polecenie z parametrem LoadUserProfile załadujemy cały profil wskazanego użytkownika (w przeciwieństwie do runas Start-Process nie ładuje profilu).

Jeżeli chcemy natomiast uruchomić polecenie z uprawnieniami administratora należy dodać w poleceniu parametr -Verb:

Start-Process notepad -Verb runAs

Zostaniemy poproszeni o podanie poświadczeń i aplikacja będzie uruchomiona z najwyższymi uprawnieniami.

2. Instalacja/odinstalowanie składników systemu Windows.

Od wersji Windows 8 system zrzuca swój pierwotny obraz do folderu instalacji. Dzięki takiemu rozwiązaniu użytkownik, widząc że system nie działa prawidłowo, może podjąć próbę przywrócenia pełnej sprawności działania swojego sprzętu poprzez wykonanie ponownej instalacji

Windows (zachowując bądź nie swoje aplikacje i dane). Działanie to nie wymaga płyty bądź nośnika instalacyjnego (jak to miało miejsce we wcześniejszych wersjach).
Prócz powyższego, dzięki temu rozwiązaniu użytkownik zawsze ma możliwość zainstalowania/odinstalowania funkcji i składników systemu Windows.

W przypadku standardowej linii poleceń będzie do narzędzie o nazwie:

dism.exe

od Deployment Image Servicing and Management. Narzędzie umożliwia zarówno instalację, kasowanie oraz przeinstalowanie wybranych składników Windows, jak i podmienianie składników systemu na nowsze wersje (szczególnie ważne w systemie Windows 10, którego wersje wychodzą dwa razy do roku). W tym wypadku mamy także możliwość tworzenia nowych obrazów systemu, które można wykorzystać do instalacji na nowych maszynach (taki obraz może zawierać, prócz dobranych już składników, także aktualizacje pobrane z Windows Update). Ze względu na mnogość opcji polecenia wskazane zostaną jedynie przykłady możliwości narzędzia:

a) sprawdzenie wszystkich dostępnych dodatków i właściwości systemu Windows:

Dism /online /Get-Features

Powyższe polecenie wyświetli wszystkie składniki dostępne dla aktualnie używanego systemu Windows (parametr Online).

Jeżeli chcemy zobaczyć zainstalowane/dostępne komponenty systemu z innej lokalizacji, wpisujemy następujące polecenie:

Dism /Image:C:\test\offline /Get-Features

gdzie ścieżka C:\test\offline może zawierać zamontowany obraz esd/wim lub dysk, na którym zainstalowany jest system Windows (należy pamiętać, że dyski twarde można montować także jako katalogi – niekoniecznie jako litery).

Jeżeli chcemy dowiedzieć się czegoś więcej na temat jednego ze składników wydajemy polecenie (zakładając, że chodzi o obecnie uruchomiony system):

dism /online /get-featureinfo /featurename:Microsoft-Hyper-V

```
C:\WINDOWS\system32>dism /online /get-featureinfo /featurename:Microsoft-Hyper-V

Deployment Image Servicing and Management tool
Version: 10.0.17763.1

Image Version: 10.0.17763.253

Feature Information:

Feature Name : Microsoft-Hyper-V
Display Name : Platforma Hyper-V
Description : Udostępnia usługi umożliwiające tworzenie maszyn wirtualnych i ich zasobów oraz zarządzanie nimi.
Restart Required : Possible
State : Disabled

Custom Properties:

(No custom properties found)

The operation completed successfully.
```

Dowiadujemy się nazwy składnika, nazwy wyświetlanej, opis, czy wymagany jest restart systemu (w większości opcji będzie status prawdopodobny/możliwy) oraz statusu (na obecnym systemie opcja jest wyłączona – Disabled).

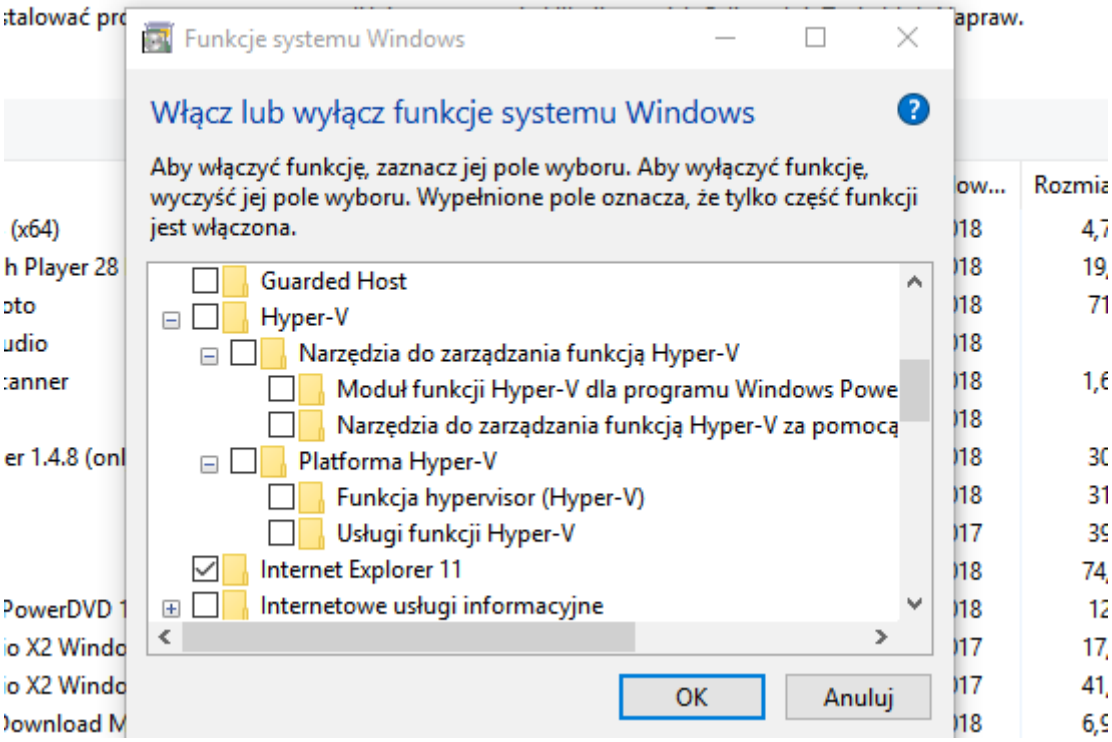
b) Jeżeli chcemy włączyć składnik, należy wydać polecenie:

Dism /online /Enable-Feature /FeatureName:Microsoft-Hyper-V

lub

Dism /online /Enable-Feature /FeatureName:Microsoft-Hyper-V /All

opcja z parametrem /All powoduje, że narzędzie zainstaluje usługę oraz wszystkie jej towarzyszące dodatki; przeważnie dodatki posiadają elementy zależne – w graficznej wersji drzewko Hyper-V prezentuje się następująco:



UWAGA: jeżeli dism nie odnajdzie pakietu źródłowego danego składnika w obecnym obrazie systemu operacyjnego, zacznie poszukiwania w usłudze Windows Update. Aby temu zapobiec można należy dodać parametr /LimitAccess (zapobiegający przed łączeniem do serwerów Microsoft). Dodatkowo możemy wskazać skąd usługa ma zostać zainstalowana (parametr /source). Przykład:

Dism /Online /Enable-Feature /FeatureName:TFTP /Source:Z:\sources\SxS /Source:C:\test\mount\windows /LimitAccess

W powyższym wypadku narzędzie otrzymuje dostęp do dwóch źródeł – na wypadek, gdyby jedno z nich nie posiadało dodatku tftp/miało nowszą wersję od poprzedniego.

c) ostatnim ważnym poleceniem jest wyłączenie wskazanego składnika/składników z systemu. Dokonuje się tego poprzez parametr /Disable-Feature:

Dism /online /Disable-Feature /FeatureName:TFTP

jeżeli wskazana funkcja była wyłączona – nic nie zostanie zmienione. Niektóre składniki (jak wcześniej wspomniane Hyper-V) po wyłączeniu będą wymagać ponownego uruchomienia systemu. Dodatkowo należy pamiętać, że wyłączenie składnika powoduje usunięcie go z systemu – wymagana będzie ponowna instalacja. Z kolei parametr /Remove spowoduje wyłączenie funkcji, jednak pozostawi składnik „gotowy” do ponownej instalacji:

Dism /online /Disable-Feature /FeatureName:TFTP /Remove

W przypadku PowerShell składnikami zarządzać można poprzez następujące polecenia:

Get-WindowsOptionalFeature

Disable-WindowsOptionalFeature

Każde z nich jest tzw. opakowaniem (wrapper) dla narzędzia dism. Ponadto PowerShell zawiera znacznie więcej poleceń do zarządzania składnikami i obrazami systemu Windows. Jeżeli chcemy zobaczyć wszystkie polecenia, które wykorzystują narzędzie dism możemy użyć polecenia:

Get-Command -Module *dism*

Po tym poleceniu wyświetlą się wszystkie polecenia, które wykorzystują dism jako swoje źródło (jest ich w sumie 60).

a) Get-WindowsOptionalFeature – pokazuje dostępne składniki systemu Windows. Wymaga co najmniej jednego parametru do poprawnego działania (-Online). Należy pamiętać, że polecenie można wydać również dla zamontowanego (nieaktywnego) systemu Windows np. poprzez wskazanie ścieżki do niego (-Path).

Przykład:

Get-WindowsOptionalFeature – Online

lub

Get-WindowsOptionalFeature -Online -FeatureName *hyper*

powyższe polecenie wyświetli nam tylko dodatki, które związane są z Hyper-V

b) Enable-WindowsOptionalFeature – włącza wybrany składnik do działania w systemie Windows.

Get-WindowsOptionalFeature -Online -FeatureName Microsoft-Hyper-V

włącza wskazany komponent. Polecenie z dodoatkowym parametrm -All

Get-WindowsOptionalFeature -Online -FeatureName Microsoft-Hyper-V -All

działa jak dism z podobnym parametrem. Analogicznie zachowa się polecenie:

Get-WindowsOptionalFeature -Online -FeatureName *hyper*

Polecenia w PowerShell można łączyć. W poniższym wypadku efekt będzie identyczny jak w przypadku poprzednio wskazanego polecenia:

Get-WindowsOptionalFeature -Online -FeatureName *hyper* | Enable-WindowsOptionalFeature -Online

Któregokolwiek polecenia nie użyjemy (za wyjątkiem pierwszego), nasz system będzie posiadał zainstalowane wszystkie składniki usługi Hyper-V (wraz z odpowiednimi narzędziami).

c) Disable-WindowsOptionalFeature – polecenie to wyłącza wskazaną/wskazane usługi w systemie Windows. Polecenie działa analogicznie do polecenia dism. Przykładowo:

Disable-WindowsOptionalFeature –Online –FeatureName "Hearts"

wyłącza z naszego systemu składnik gry karcianej (w pełni go usuwając). Natomiast polecenie:

Disable-WindowsOptionalFeature –Online –FeatureName "Hearts" -Remove

pozostawia jego plik konfiguracyjny (manifest), dzięki czemu składnik może być w dowolnej chwili przywrócony do systemu (i będzie działać tak jak został skonfigurowany). Jednak dzięki chwilowemu usunięciu jego instalacji wynikowy obraz systemu jest mniejszy.

3. Zarządzanie dyskami.

Domyślnym narzędziem zarządzania dyskami w konsoli jest diskpart. Narzędzie to pozwala na inicjowanie, partycjonowanie, przepartycjonowanie oraz łączenie dysków (LVM) i partycji. Dzięki niemu możemy odczytać informacje o podłączonych dyskach, partycjach i wolumenach. Aby skorzystać z narzędzia w konsoli wpisujemy polecenie:

diskpart

Narzędzie zaprojektowane jest względem nowszych standardów zarządzania. Posiada ono mianowicie poziomy, na których obecnie się znajdujemy. Przykładowo zaraz po uruchomieniu narzędzie posiada pewien zestaw komend, które może wydać użytkownik. Po wybraniu komendy, poprzez separator (jest nim spacja) możliwe jest doprecyzowanie polecenia. Z kolei kolejna część polecenia może posiadać dodatkowe podpolecenie. Przykładowo:

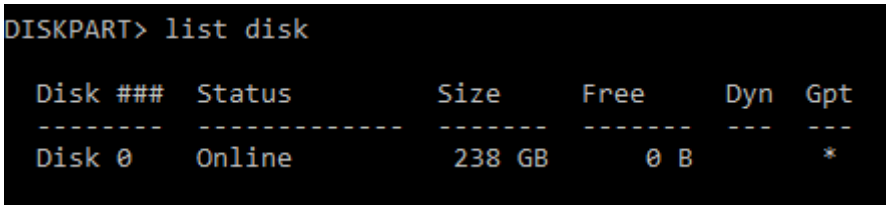
LIST

będzie poleceniem, które można połączyć z następującymi podpoleceniami:

DISK, VOLUME, PARTITION, VDISK

Przykładowo:

LIST DISK



Ponieważ mamy jeden dysk, możemy wybrać go poprzez polecenie:

SELECT DISK 0

Od tego momentu możemy na wskazanym dysku m. in.:

- powiększać poszczególne partycje (EXPAND, EXTEND)
- pomniejszać partycje (SHRINK)
- wskazywać aktywne partycje (ACTIVE)
- formatować wskazany dysk/partycje (FORMAT)

Lista poleceń jest znacznie dłuższa, wpisując znak zapytania (?) i klikając [ENTER] LUB polecenie i znak zapytania [ENTER] otrzymamy informację o wszystkich dostępnych poleceniach w danym poziomie narzędzia.
Przykład wyświetlenia pomocy:

```
DISKPART> list ?

Microsoft DiskPart version 10.0.17763.1

DISK        - Display a list of disks. For example, LIST DISK.
PARTITION   - Display a list of partitions on the selected disk.
              For example, LIST PARTITION.
VOLUME      - Display a list of volumes. For example, LIST VOLUME.
VDISK       - Displays a list of virtual disks.
```

W przypadku PowerShell mamy około 51 dostępnych poleceń do obsługi przestrzeni magazynowej wszelkiego rodzaju. W tekście zostaną przedstawione te najważniejsze, pozostałe można pozyskać poprzez Get-Command/Get-Help.

a) Get-Disk – otrzymamy informacje o wszystkich dostępnych, w ramach urządzenia, magazynach danych. Informacje można otrzymać o wskazanym urządzeniu poprzez podanie np. jego nazwy, umiejscowienia na magistrali. Ponadto wyjście również można formatować wedle własnego uznania.

b) Get-Partition – analogicznie do poprzedniego polecenia otrzymamy informacje dotyczące partycji dostępnych w systemie operacyjnym.

c) Format-Volume – pozwala na sformatowanie wybranego dysku. Przykładowo:

Format-Volume -DriveLetter C -FileSystem FAT32 -FullFormat –Force

formatuje dysk z literą C na system plików FAT32 i wymusza pełne formatowanie (wolne, dokładne).

d) Set-Volume – pozwana na dokonanie zmian litery dysku, etykiety magazynu itp.

e) Set-Disk – pozwala na ustawienie początkowych wartości dla magazynu danych. Poprzez opcję można przykładowo ustawić styl partycji (MBR, GPT) czy aktywność dysku (Online, Offline)

f) Set-Partition – pozwala na ustawienie wartości dla wyszczególnionej (wyszczególnionych) partycji. Ustawienia obejmują zmianę litery dysku/katalogu montowania, aktywności partycji

(aktywna partycja może być wykorzystana do uruchamiania systemu operacyjnego – MBR), jak i przesunięcia jej względem początku dysku (przesunięcie podawane w bajtach).

4. Zarządzanie siecią

Domyślnie do sprawdzenia adresu IP w linii poleceń używa się polecenia ipconfig. Polecenie to pozwala także na włączenie/wyłączenie sieci, nadanie nowego adresu IP dla interfejsu(ów) itp.

System Windows od wersji 2000 posiada również specjalne narzędzie o nazwie netsh (Network Shell). Pozwala ono na zarządzanie ustawieniami sieciowymi systemu Windows. Możemy zmieniać ustawienia zapory sieciowej (advfirewall), interfejsów sieciowych (interface) jak i profilami sieciowymi (np. wlan w przypadku profili sieci bezprzewodowej). W systemach z serii 10 popularnym, lecz niekoniecznie w pozytywnym aspekcie, poleceniem jest restartowanie działania gniazd sieciowych (często raportowany błąd użytkowników, który, pomimo braku zmian w konfiguracji sieciowej systemu oraz urządzeń, powoduje utratę połączenia do usług sieciowych). Polecenie to ma postać:

netsh winsock reset

Powyższe polecenie zresetuje wszystkie ustawienia połączeń sieciowych systemu Windows. By w pełni osiągnąć efekt zawsze wymagane jest ponowne uruchomienie.

Narzędzie może wykonać jednorazową operację (jak powyższe, uruchamiające wyczyszczenie konfiguracji gniazd sieciowych) lub może zostać uruchomione jako konsola, w której możliwe jest wydawanie wielu niezależnych poleceń. Powłokę (konsolę) uaktywnia się poprzez wpisanie w wierszu poleceń nazwy narzędzia:

netsh

Użytkowanie narzędzia jest analogiczne do diskpart. Każdy poziom menu można prześledzić wpisując w konsolę help/znak zapytania (?) i naciskając [ENTER]. Pomiędzy poziomami (kategoriami) przełącza się poprzez podanie jednej z nich, np. interface, advfirewall, wlan. Z kolei powrót do kategorii wyżej następuję po wpisaniu dwóch kropek (..). W przeciwieństwie do większości narzędzi, koniec pracy następuje po wpisaniu słowa bye (na wzór większości urządzeń sieciowych).

Ponieważ narzędzie posiada bardzo rozbudowaną funkcjonalność, pokazane zostaną jedynie wybrane możliwości.

a) ustawienie adresu IP dla wskazanego interfejsu (wraz z maską, bramą oraz serwerem DNS):

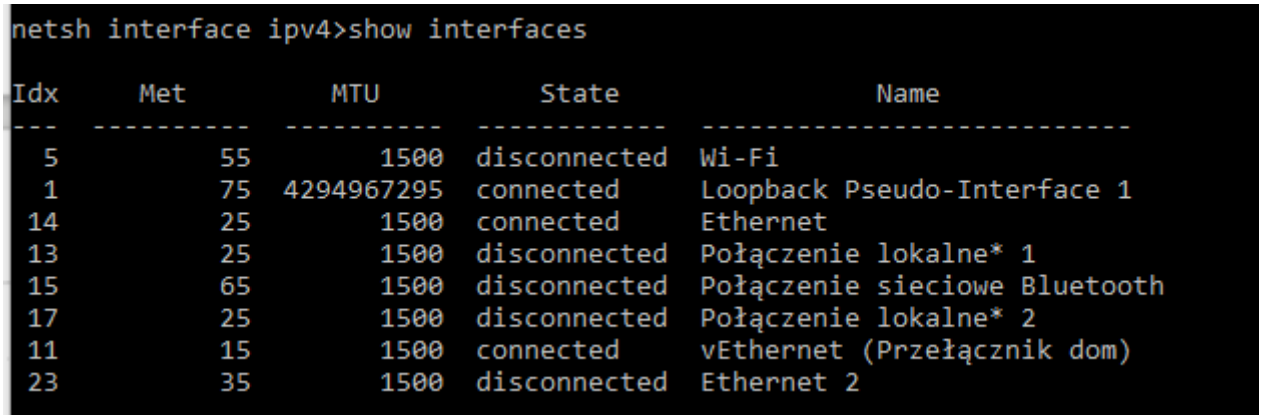
na początek musimy doprecyzować, którą wersję protokołu chcemy ustawić. Zakładamy, nadal niestety popularną, wersję 4. W związku z tym wydajemy polecenie:

netsh interface ipv4

Pierwszym krokiem będzie pozyskanie nazwy naszego interfejsu (dla którego chcemy ustawić adresy). Wydajemy polecenie (po wydaniu poprzedniego):

show interfaces

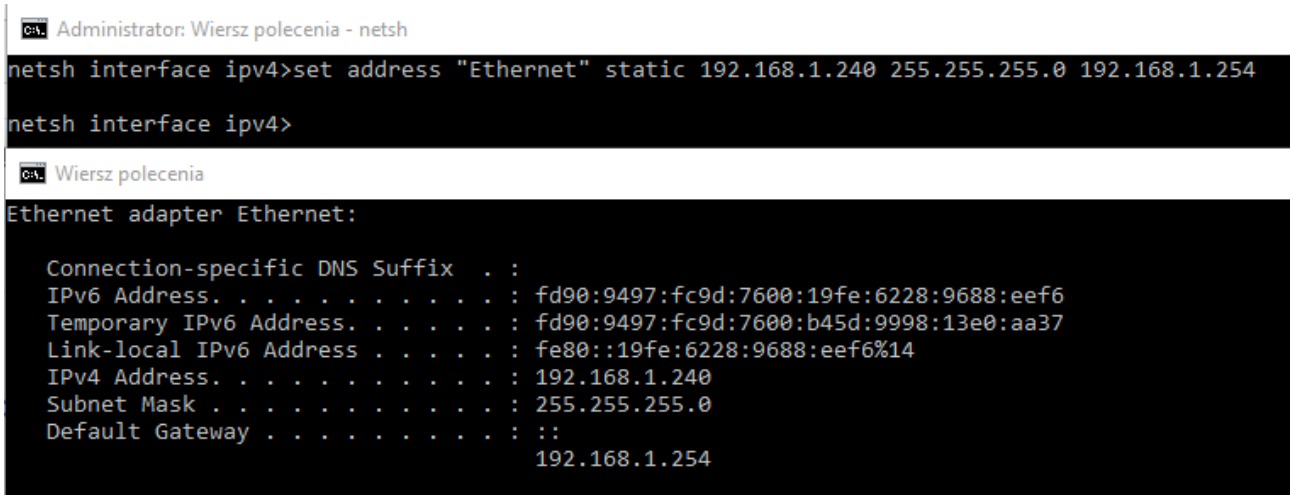
Wynik może być podobny do poniższego:



Interfejsy podłączone mają statu Connected. Załóżmy, że wybieramy interfejs Ethernet. Na początek ustawimy adres IP z puli 192.168.1.0/24 i bramą 192.168.1.254.

set address "Ethernet" static 192.168.1.240 255.255.255.0 192.168.1.254

Efekt działania (po wydaniu w drugim wierszu poleceń komendy ipconfig):



alternatywne ustawienie adresu:

set address "Ethernet" static 192.168.1.240/24 gateway=192.168.1.254

lub

set address name="Ethernet" source=static address=192.168.1.240 mask=255.255.255.0 gateway=192.168.1.254

pełną listę możliwości polecenia (wraz z przykładami użycia) można poznać po wpisaniu w konsolę

set address ? [ENTER]

Celem sprawdzenia konfiguracji bezpośrednio w netsh można wydać polecenie (cały czas w gałęzi netsh interface ipv4):

show address name=”Ethernet”

```
netsh interface ipv4>show address name="Ethernet"

Configuration for interface "Ethernet"
    DHCP enabled:                         No
    IP Address:                           192.168.1.240
    Subnet Prefix:                        192.168.1.0/24 (mask 255.255.255.0)
    Default Gateway:                      192.168.1.254
    Gateway Metric:                       1
    InterfaceMetric:                      25
```

Dodanie dwóch lub większej ilości serwerów DNS możliwe jest dzięki następującym poleceniom:

set dnsservers name="Ethernet" source=static address=1.1.1.1 register=primary

powyższe polecenie dodaje do interfejsu Ethernet nowy, statyczny adres serwera DNS 1.1.1.1 (usługa Cloudflare). DNS zostaje zarejestrowany jako główny. Jeżeli chcemy dodać kolejny adres DNS (np. Google) trzeba użyć nieco innego polecenia:

add dnsservers name="Ethernet" address=8.8.8.8 index=2

polecenie DODAJE (add – nie set) adres 8.8.8.8 dla interfejsu Ethernet. Ponadto wskazujemy, który na liście DNS ma być dodawany adres (w tym wypadku jako drugi). Na koniec możemy sprawdzić efekt naszych ustawień:

```
netsh interface ipv4>set dnsservers name="Ethernet" source=static address=1.1.1.1 register=primary

netsh interface ipv4>add dnsservers name="Ethernet" address=8.8.8.8 index=2

netsh interface ipv4>show dnsservers name="Ethernet"

Configuration for interface "Ethernet"
    Statically Configured DNS Servers:    1.1.1.1
                                          8.8.8.8
    Register with which suffix:           Primary only
```

Gdyby z jakiegoś powodu nie zadziałał nam dostęp do sieci (przy założeniu, że brama posiada dostęp do sieci WAN), należy dodać dodatkową drogę do trasowania pakietów:

add route 0.0.0.0/0 "Ethernet" 192.168.1.254 0 0 yes

Od tego momentu powinniśmy mieć dostęp do sieci rozległej (WAN).

Oczywiście tak jak w poprzednich przypadkach, PowerShell również możne znacząco ułatwić zarządzanie siecią w systemie Windows. Opisywane ustawienie powyżej w PowerShell wyglądało by następująco:

a) Podobnie jak to miało miejsce w przypadku netsh, musimy wiedzieć jaką nazwę posiada nasz interfejs. Na poniższym zrzucie ekranu podświetlony został właściwy dla nas interfejs. Proszę zwrócić uwagę na fakt iż interfejs przewija się tutaj dwa razy – dla IPv6 oraz IPv4. Jeżeli chcielibyśmy wyświetlić interfejsy tylko połączeń IPv4, należy wydać polecenie:

Get-NetIPInterface -AddressFamily IPv4

Bez podania rodziny adresacji wynik widać na zrzucie:

```
PS C:\WINDOWS\system32> Get-NetIPInterface

ifIndex InterfaceAlias                 AddressFamily NlMtu(Bytes) InterfaceMetric Dhcp     ConnectionState PolicyStore
------- --------------                 ------------- ------------ --------------- ----     --------------- -----------
23      Ethernet 2                     IPv6                  1500              35 Enabled  Disconnected    ActiveStore
11      vEthernet (Przełącznik dom)    IPv6                  1500              15 Enabled  Connected       ActiveStore
17      Połączenie lokalne* 2          IPv6                  1500              25 Disabled Disconnected    ActiveStore
15      Połączenie sieciowe Bluetooth  IPv6                  1500              65 Disabled Disconnected    ActiveStore
13      Połączenie lokalne* 1          IPv6                  1500              25 Enabled  Disconnected    ActiveStore
14      Ethernet                       IPv6                  1500              25 Enabled  Connected       ActiveStore
1       Loopback Pseudo-Interface 1    IPv6            4294967295              75 Disabled Connected       ActiveStore
12      Teredo Tunneling Pseudo-Inte... IPv6                 1280              75 Enabled  Connected       ActiveStore
23      Ethernet 2                     IPv4                  1500              35 Enabled  Disconnected    ActiveStore
11      vEthernet (Przełącznik dom)    IPv4                  1500              15 Enabled  Connected       ActiveStore
17      Połączenie lokalne* 2          IPv4                  1500              25 Enabled  Disconnected    ActiveStore
15      Połączenie sieciowe Bluetooth  IPv4                  1500              65 Enabled  Disconnected    ActiveStore
13      Połączenie lokalne* 1          IPv4                  1500              25 Enabled  Disconnected    ActiveStore
14      Ethernet                       IPv4                  1500              25 Disabled Connected       ActiveStore
1       Loopback Pseudo-Interface 1    IPv4            4294967295              75 Disabled Connected       ActiveStore
5       Wi-Fi                          IPv4                  1500              55 Enabled  Disconnected    ActiveStore
```

b) Teraz, mając już nazwę interfejsu (Ethernet) musimy ustawić nowy adres. Wykonanie poniższego polecenia powinno zmienić adres na nowy:

New-NetIPAddress -InterfaceIndex 14 -IPAddress 192.168.1.220 -PrefixLength 24 -DefaultGateway 192.168.1.1 -WhatIf

wynik poniżej:

```
PS C:\WINDOWS\system32> New-NetIPAddress -InterfaceIndex 14 -IPAddress 192.168.1.220 -PrefixLength 24 -DefaultGateway 192.168.1.1 -WhatIf
What if: Performing operation "New" on Target "NetIPAddress -IPv4Address 192.168.1.220 -InterfaceIndex 14 -Store Active"
What if: Performing operation "New" on Target "NetRoute -DestinationPrefix 0.0.0.0/0 -InterfaceIndex 14 -NextHop 192.168.1.1 -Store Active"
What if: Performing operation "New" on Target "NetIPAddress -IPv4Address 192.168.1.220 -InterfaceIndex 14 -Store Persistent"
What if: Performing operation "New" on Target "NetRoute -DestinationPrefix 0.0.0.0/0 -InterfaceIndex 14 -NextHop 192.168.1.1 -Store Persistent"
PS C:\WINDOWS\system32>
```

W tym wypadku należy zwrócić uwagę na parametr WhatIf. Spora ilość poleceń PowerShell pozwala na jego dodanie. Powoduje on, że polecenie nie wykonuje się naprawdę, a administrator/użytkownik otrzymuje jedynie informacje co by się stało, gdyby polecenie zostało realnie wykonane.

c) ustawiamy serwer DNS:

Set-DnsClientServerAddress -InterfaceIndex 14 -ServerAddresses 8.8.8.8,192.168.1.254

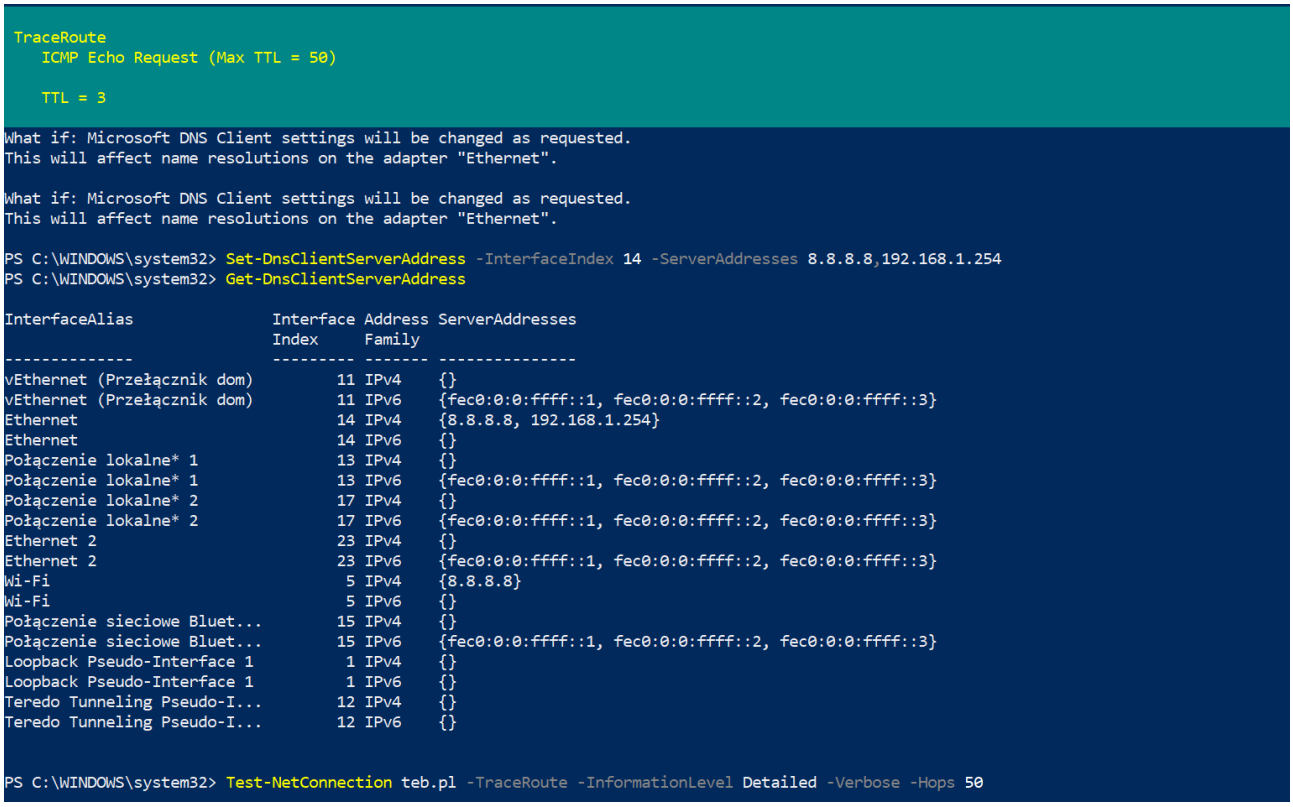
```
PS C:\WINDOWS\system32> Set-DnsClientServerAddress -InterfaceIndex 14 -ServerAddresses 8.8.8.8,192.168.1.254
PS C:\WINDOWS\system32> Get-DnsClientServerAddress

InterfaceAlias               Interface Address ServerAddresses
                             Index     Family
--------------               --------- ------- ---------------
vEthernet (Przełącznik dom)         11 IPv4    {}
vEthernet (Przełącznik dom)         11 IPv6    {fec0:0:0:ffff::1, fec0:0:0:ffff::2, fec0:0:0:ffff::3}
Ethernet                            14 IPv4    {8.8.8.8, 192.168.1.254}
Ethernet                            14 IPv6    {}
Połączenie lokalne* 1               13 IPv4    {}
Połączenie lokalne* 1               13 IPv6    {fec0:0:0:ffff::1, fec0:0:0:ffff::2, fec0:0:0:ffff::3}
Połączenie lokalne* 2               17 IPv4    {}
Połączenie lokalne* 2               17 IPv6    {fec0:0:0:ffff::1, fec0:0:0:ffff::2, fec0:0:0:ffff::3}
Ethernet 2                          23 IPv4    {}
```

d) sprawdzamy łączność do sieci rozległej; użyjemy kolejnego polecenia PowerShell zamiast ping:

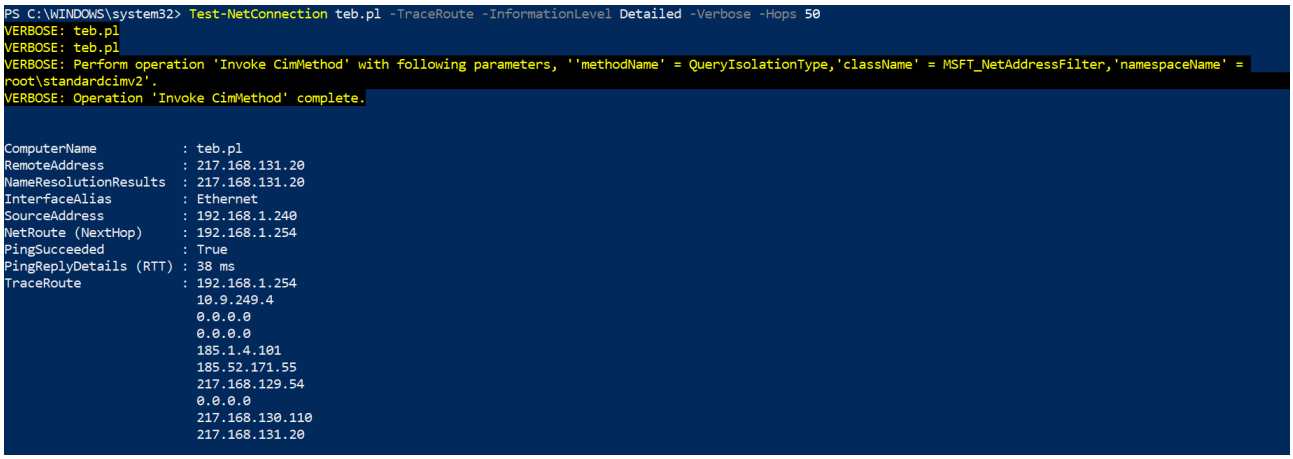
Test-NetConnection teb.pl -TraceRoute -InformationLevel Detailed -Verbose -Hops 50

Funkcja sprawdza łączność pomiędzy użytkowaną maszyną a adresem IP/nazwą domenową wskazaną w parametrze. Dodatkowo pozwala prześledzić ścieżkę pakietu do miejsca docelowego (traceroute). Parametr informationlevel ustawiony został na szczegóły (więcej informacji o

połączeniu). Verbose z kolei pozwoli na poznanie wszelkich komentarzy, co wykonuje polecenie. Ostatni parametr, Hops pozwala na zmianę ilości przeskoków, po których polecenie uzna, że pakiet nie może dotrzeć do miejsca docelowego (domyślnie narzędzie operuje na wartości 30 przeskoków). W trakcie wykonania narzędzie będzie wyświetlało w górnej części konsoli ramkę informująca o kolejnych postępach:



Ostateczny wynik działania:



Oczywiście narzędzie można wywołać w postaci prostej, tj.

Test-NetConnection teb.pl

dostaniemy wtedy informacje o średnim czasie dostępu do adresu docelowego, adresach źródłowych/docelowych jak i interfejsie, który brał udział w komunikacji.

e) gdyby z jakiejś przyczyny pakiety nie przeszły do docelowego miejsca może to oznaczać, że ustawiliśmy zły adres lub ścieżka pakietów została źle ustawiona. Ścieżki można sprawdzić poleceniem:

Get-NetRoute

Jeżeli nie będzie tam ścieżki z adresem docelowym 0.0.0.0/0, którego przeskokiem będzie nasza brama LUB metryka będzie większa od 0, trzeba na nowo dodać naszą ścieżkę. Jeżeli ścieżka istnieje, trzeba ją usunąć:

Remove-NetRoute -InterfaceIndex 14 -DestinationPrefix 0.0.0.0/0 -Confirm:$true

```
PS C:\WINDOWS\system32> Remove-NetRoute -InterfaceIndex 14 -DestinationPrefix 0.0.0.0/0 -Confirm:$true

Confirm
Are you sure you want to perform this action?
Performing operation "Remove" on Target "NetRoute -DestinationPrefix 0.0.0.0/0 -InterfaceIndex 14 -NextHop 192.168.1.254 -Store Active"
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is "Y"):

Confirm
Are you sure you want to perform this action?
Performing operation "Remove" on Target "NetRoute -DestinationPrefix 0.0.0.0/0 -InterfaceIndex 14 -NextHop 192.168.1.254 -Store Persistent"
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is "Y"): Y
PS C:\WINDOWS\system32>
```

Następnie dodać ją na nowo (bądź utworzyć):

New-NetRoute -InterfaceIndex 14 -DestinationPrefix 0.0.0.0/0 -NextHop 192.168.1.254 -RouteMetric 0

Bezwzględnie należy pamiętać, by ścieżka ta miała najwyższy priorytet (zero). W przeciwnym wypadku pakiety kierowane poza sieć mogą zgubić się w połączeniach lokalnych. Internet powinien bez problemu zacząć działać (o ile ustawiliśmy odpowiednie adresy IP).

5. Pozostałe wybrane operacje, z którymi warto się zapoznać.

Posiadając wersję 5.x PowerShell można w prosty sposób wykonać te same operacje, które w normalnych warunkach trzeba byłoby wykonywać systemowymi, niekoniecznie jednoznacznie brzmiącymi, poleceniami. PowerShell ma tę zaletę, że każdy może go rozwijać wedle swojego upodobania, dokładać bądź zmieniać w nim funkcje, które następnie można w dowolnym momencie przemieścić na inne komputery (pisanie prostych funkcji i skryptów będzie niezależnym tematem).

Na tę chwilę można zobaczyć, że każdą z mniej lub bardziej skomplikowanych operacji możemy wykonać poprzez intuicyjnie nazwane polecenie PowerShell. Jeżeli z jakiegoś powodu zapomnimy nazwy polecenia, zawsze możemy wpisać:

Get-Command *nazwa*

a wyświetlą nam się skojarzone z określonym słowem polecenia – włączając w to narzędzia systemowe (wpisując słowo *calc* wyświetli nam się także aplikacja calc.exe)

```
PS C:\WINDOWS\system32> Get-Command *calc*

CommandType     Name                                               Version    Source
-----------     ----                                               -------    ------
Function        Get-DscLocalConfigurationManager                   1.1        PSDesiredStateConfiguration
Cmdlet          Set-DscLocalConfigurationManager                   1.1        PSDesiredStateConfiguration
Application     calc.exe                                           10.0.17... C:\Windows\system32\calc.exe
```

Oznacza to, że nie musimy nawet zapamiętać dokładnej nazwy polecenia/nazwy programu, a jedynie jej fragment. Z kolei

Get-Help Test-NetConnection

wyświetli nam odpowiednią treść pomocy dla wskazanego przez nas polecenia. W domyślnej, świeżo zainstalowanej wersji otrzymamy jedynie informacje o wszystkich parametrach polecenia, jednak przy pierwszym uruchomieniu Get-Help PowerShell zapyta nas czy chcemy pobrać dokumentację na nasz komputer. Po zatwierdzeniu otrzymamy możliwość wyświetlenia szczegółów dotyczących polecenia:

Get-Help Test-NetConnection -Detailed

bądź, jeżeli znamy już jego działanie, a potrzebujemy jedynie przypomnienia jego wywołania:

Get-Help Test-NetConnection -Examples

Konsola ma też niepodważalną zaletę podpowiadania składni – wystarczy naciskać [TAB] by poznać kolejne (ułożone alfabetycznie) polecenia, a po wpisaniu polecenia i myślnika można także poznać wszystkie parametry. To kolejne udogodnienie względem administratorów, którzy nie muszą już zapamiętywać wszystkich poleceń.

Poniżej część poleceń, jakie mogą okazać się przydatne w codziennej pracy administratora systemu Windows (bądź na egzaminie EE08):

a) aktualna wersja PowerShell:

echo $PSVersionTable

b) pobranie informacji o aktywnych użytkownikach;

quser.exe

c) pobieranie informacji o systemie Windows, celem późniejszej modyfikacji/pobierania informacji o systemie i jego komponentach:

Get-WmiObject -List

Jeżeli chcemy pobrać informacje o konkretnej klasie, np. Win32:

Get-WmiObject -List *Win32*

Załóżmy, że zainteresowała nas klasa Win32_SystemBIOS. Chcąc dowiedzieć się co przechowuje, możemy wydać następujące polecenie:

(Get-WmiObject Win32_SystemBIOS)

Wyświetlą nam się wszystkie obiekty, jakie przechowuje wskazana klasa:

```
PS C:\WINDOWS\system32> (Get-WmiObject Win32_SystemBIOS)


__GENUS          : 2
__CLASS          : Win32_SystemBIOS
__SUPERCLASS     : CIM_SystemComponent
__DYNASTY        : CIM_Component
__RELPATH        : Win32_SystemBIOS.GroupComponent="\\\\DESKTOP-TOI1II2\\root\\cimv2:Win32_ComputerSystem.Name=\"DESKTOP-TOI1II2\"",PartComponent="\\\\DESKTOP-TOI1II2\\root
                   \\cimv2:Win32_BIOS.Name=\"R0DET86W (1.86 )\",SoftwareElementID=\"R0DET86W (1.86 )\",SoftwareElementState=3,TargetOperatingSystem=0,Version=\"LENOVO - 860
                   \""
__PROPERTY_COUNT : 2
__DERIVATION     : {CIM_SystemComponent, CIM_Component}
__SERVER         : DESKTOP-TOI1II2
__NAMESPACE      : root\cimv2
__PATH           : \\DESKTOP-TOI1II2\root\cimv2:Win32_SystemBIOS.GroupComponent="\\\\DESKTOP-TOI1II2\\root\\cimv2:Win32_ComputerSystem.Name=\"DESKTOP-TOI1II2\"",PartCompone
                   nt="\\\\DESKTOP-TOI1II2\\root\\cimv2:Win32_BIOS.Name=\"R0DET86W (1.86 )\",SoftwareElementID=\"R0DET86W (1.86 )\",SoftwareElementState=3,TargetOperatingSy
                   stem=0,Version=\"LENOVO - 860\""
GroupComponent   : \\DESKTOP-TOI1II2\root\cimv2:Win32_ComputerSystem.Name="DESKTOP-TOI1II2"
PartComponent    : \\DESKTOP-TOI1II2\root\cimv2:Win32_BIOS.Name="R0DET86W (1.86 )",SoftwareElementID="R0DET86W (1.86 )",SoftwareElementState=3,TargetOperatingSystem=0,Versi
                   on="LENOVO - 860"
PSComputerName   : DESKTOP-TOI1II2
```

Teraz chcąc np. wybrać poszczególne informacje (i je w jakiś sposób wykorzystać) można pobawić się w przekazywanie wskazanej zmiennej do innych poleceń. Przykładowo chcemy czytelnie wyświetlić zawartość __PATH. Możemy przekonwertować zachowane w niej wartości poprzez rozdzielenie ich względem znaku i/lub sekwencji znaków na mniejsze ciągi znakowe. Polecenie:

(Get-WmiObject Win32_SystemBIOS).__PATH | ConvertFrom-String -Delimiter ','

spowoduje wyświetlenie:

```
PS C:\WINDOWS\system32> (Get-WmiObject Win32_SystemBIOS).__PATH | ConvertFrom-String -Delimiter ','


P1 : \\DESKTOP-TOI1II2\root\cimv2:Win32_SystemBIOS.GroupComponent="\\\\DESKTOP-TOI1II2\\root\\cimv2:Win32_ComputerSystem.Name=\"DESKTOP-TOI1II2\""
P2 : PartComponent="\\\\DESKTOP-TOI1II2\\root\\cimv2:Win32_BIOS.Name=\"R0DET86W (1.86 )\"
P3 : SoftwareElementID=\"R0DET86W (1.86 )\"
P4 : SoftwareElementState=3
P5 : TargetOperatingSystem=0
P6 : Version=\"LENOVO - 860\""
```

d) podobnymi poleceniami cechuje się Cim (Common Information Model). Pozwala on na wynajdywanie szczegółowych informacji dotyczących systemu, zainstalowanego oprogramowania, wersji systemu operacyjnego itp. Polecenie działa podobnie do poleceń Wmi. Przykładowo by znaleźć odpowiednią klasę Cim należy wydać polecenie:

Get-CimClass

dostaniemy wówczas informacje o wszystkich dostępnych klasach Cim. Załóżmy, że zainteresowała nas klasa CIM_OperatingSystem. Domyślnie możemy wywołać polecenie:

Get-CimInstance CIM_OperatingSystem

```
PS C:\WINDOWS\system32> Get-CimInstance CIM_OperatingSystem

SystemDirectory     Organization BuildNumber RegisteredUser SerialNumber            Version
---------------     ------------ ----------- -------------- ------------            -------
C:\WINDOWS\system32               17763       Windows User   00330-50000-00000-AAOEM 10.0.17763
```

Powyższa odpowiedź nie jest jednak pełna. Zawiera ona bowiem w sobie znacznie więcej treści (jak chociażby datę ostatniego logowania, datę instalacji systemu itp.). Aby wyciągnąć pełne informacje trzeba użyć małej sztuczki. PowerShell potrafi przemieniać odpowiedzi w format CSV/zamieniać na format CSV. Zamieniając klasę na CSV przemieścimy do niej WSZYSTKIE zawarte w niej informacje:

Get-CimInstance CIM_OperatingSystem | ConvertTo-Csv

```
PS C:\WINDOWS\system32> Get-CimInstance CIM_OperatingSystem | ConvertTo-Csv
#TYPE Microsoft.Management.Infrastructure.CimInstance#root/cimv2/Win32_OperatingSystem
"Caption","Description","InstallDate","Name","Status","CreationClassName","CSCreationClassName","CSName","CurrentTimeZone","Distributed","FreePhysicalMemory","FreeSpaceInPa
ingFiles","FreeVirtualMemory","LastBootUpTime","LocalDateTime","MaxNumberOfProcesses","MaxProcessMemorySize","NumberOfLicensedUsers","NumberOfProcesses","NumberOfUsers","OS
ype","OtherTypeDescription","SizeStoredInPagingFiles","TotalSwapSpaceSize","TotalVirtualMemorySize","TotalVisibleMemorySize","Version","BootDevice","BuildNumber","BuildType
,"CodeSet","CountryCode","CSDVersion","DataExecutionPrevention_32BitApplications","DataExecutionPrevention_Available","DataExecutionPrevention_Drivers","DataExecutionPreven
ion_SupportPolicy","Debug","EncryptionLevel","ForegroundApplicationBoost","LargeSystemCache","Locale","Manufacturer","MUILanguages","OperatingSystemSKU","Organization","OSA
chitecture","OSLanguage","OSProductSuite","PAEEnabled","PlusProductID","PlusVersionNumber","PortableOperatingSystem","Primary","ProductType","RegisteredUser","SerialNumber"
"ServicePackMajorVersion","ServicePackMinorVersion","SuiteMask","SystemDevice","SystemDirectory","SystemDrive","WindowsDirectory","PSComputerName"
"Microsoft Windows 10 Pro","","30.12.2018 14:04:59","Microsoft Windows 10 Pro|C:\WINDOWS|\Device\Harddisk0\Partition3","OK","Win32_OperatingSystem","Win32_ComputerSystem","
ESKTOP-TOI1II2","60","False","9085684","1786404","6351168","23.01.2019 03:17:27","11.02.2019 00:21:27","4294967295","137438953344","0","272","2","18",,"2490368",,"19168044"
"16677676","10.0.17763","\Device\HarddiskVolume1","17763","Multiprocessor Free","1250","48",,"True","True","True","2","False","256","2",,"0415","Microsoft Corporation","Sys
em.String[]","48","","64-bitowy","1045","256",,,,"False","True","1","Windows User","00330-50000-00000-AAOEM","0","0","272","\Device\HarddiskVolume3","C:\WINDOWS\system32","
:","C:\WINDOWS",
PS C:\WINDOWS\system32>
```

Jak można zauważyć, nie jest to dla nas zbyt czytelne (oczywiście można te dane zapisać do pliku i odczytać np. w Excelu). Chcąc jednak otrzymać pełne informacje w konsoli można ponownie przemienić otrzymane dane na format obiektowy PowerShell – tym razem możliwy do formatowania (poprzedni formatował tylko domyślnie wyświetlane wartości):

Get-CimInstance CIM_OperatingSystem | ConvertTo-Csv | ConvertFrom-Csv

Tym razem otrzymamy pełne dane dotyczące naszego systemu (poniżej fragment):

```
PS C:\WINDOWS\system32> Get-CimInstance CIM_OperatingSystem | ConvertTo-Csv | ConvertFrom-Csv


Caption                                 : Microsoft Windows 10 Pro
Description                             :
InstallDate                             : 30.12.2018 14:04:59
Name                                    : Microsoft Windows 10 Pro|C:\WINDOWS|\Device\Harddisk0\Partition3
Status                                  : OK
CreationClassName                       : Win32_OperatingSystem
CSCreationClassName                     : Win32_ComputerSystem
CSName                                  : DESKTOP-TOI1II2
CurrentTimeZone                         : 60
Distributed                             : False
FreePhysicalMemory                      : 9138100
FreeSpaceInPagingFiles                  : 1789720
FreeVirtualMemory                       : 6579900
LastBootUpTime                          : 23.01.2019 03:17:27
LocalDateTime                           : 11.02.2019 00:23:39
MaxNumberOfProcesses                    : 4294967295
MaxProcessMemorySize                    : 137438953344
NumberOfLicensedUsers                   : 0
NumberOfProcesses                       : 272
NumberOfUsers                           : 2
OSType                                  : 18
OtherTypeDescription                    :
SizeStoredInPagingFiles                 : 2490368
TotalSwapSpaceSize                      :
TotalVirtualMemorySize                  : 19168044
TotalVisibleMemorySize                  : 16677676
Version                                 : 10.0.17763
BootDevice                              : \Device\HarddiskVolume1
BuildNumber                             : 17763
BuildType                               : Multiprocessor Free
CodeSet                                 : 1250
CountryCode                             : 48
CSDVersion                              :
DataExecutionPrevention_32BitApplications : True
```

Jeżeli chcemy wyświetlić tylko niektóre informacje, przykładowo w tabeli, możemy użyć takiego polecenia:

Get-CimInstance CIM_OperatingSystem | ConvertTo-Csv | ConvertFrom-Csv | Format-Table -Property Caption,Version,InstallDate

gdzie w parametrze Property podajemy nazwy kolumn (pełne nazwy), które chcemy wyświetlić. Efekt jest następujący:

```
PS C:\WINDOWS\system32> Get-CimInstance CIM_OperatingSystem | ConvertTo-Csv | ConvertFrom-Csv | Format-Table -Property Caption,Version,InstallDate

Caption                 Version    InstallDate
-------                 -------    -----------
Microsoft Windows 10 Pro 10.0.17763 30.12.2018 14:04:59
```

Jak widać polecenia w PowerShell można łączyć by uzyskać przejrzyste i łatwe do interpretacji wyniki. Większość tutaj użytych poleceń bez kłopotu można wykorzystać przy innych poleceniach (niemal każde polecenie można konwertować co najmniej jako String, większość poleceń

PowerShell można natomiast konwertować jako CSV, formatować można dowolne wyjście polecenia PowerShell).

e) Jeżeli na szybko chcemy pobrać jakąś wartość z rejestru systemowego:

(Get-ItemProperty "HKLM:\SOFTWARE\Microsoft\Windows NT\CurrentVersion").ReleaseId

Polecenie zwróci nam informacje o wersji używanego systemu Windows:

```
PS C:\WINDOWS\system32> (Get-ItemProperty "HKLM:\SOFTWARE\Microsoft\Windows NT\CurrentVersion").ReleaseId
1809
PS C:\WINDOWS\system32>
```

f) pobranie informacji o obecnie użytkowanym sprzęcie komputerowym:

Get-ComputerInfo

Polecenie łączy w sobie większość obiektów CIM, wszystkie wyświetlane jako lista.

```
PS C:\WINDOWS\system32> Get-ComputerInfo


WindowsBuildLabEx                                   : 17763.1.amd64fre.rs5_release.180914-1434
WindowsCurrentVersion                               : 6.3
WindowsEditionId                                    : Professional
WindowsInstallationType                             : Client
WindowsInstallDateFromRegistry                      : 30.12.2018 13:04:59
WindowsProductId                                    : 00330-50000-00000-AAOEM
WindowsProductName                                  : Windows 10 Pro
WindowsRegisteredOrganization                       :
WindowsRegisteredOwner                              : Windows User
WindowsSystemRoot                                   : C:\WINDOWS
WindowsVersion                                      : 1809
BiosCharacteristics                                 :
BiosBIOSVersion                                     :
BiosBuildNumber                                     :
BiosCaption                                         :
BiosCodeSet                                         :
BiosCurrentLanguage                                 :
BiosDescription                                     :
BiosEmbeddedControllerMajorVersion                  :
```

Można je zmniejszyć poprzez użycie polecenia select, Format itd.

g) PowerShell pozwala także na pakowanie/rozpakowanie archiwów:

Compress-Archive -Path '.' -DestinationPath 'plik.zip'

Powyższe polecenie spakuje bieżącą lokalizację (wskazana jako kropka) do pliku plik.zip.

Poleceniem przeciwnym jest

Expand-Archive

h) przy użyciu konsoli możemy także pobierać zawartość zarówno folderów otoczenia sieciowego jaki i ze stron WWW:

Na początek spróbujmy pobrać zawartość dowolnej strony WWW:

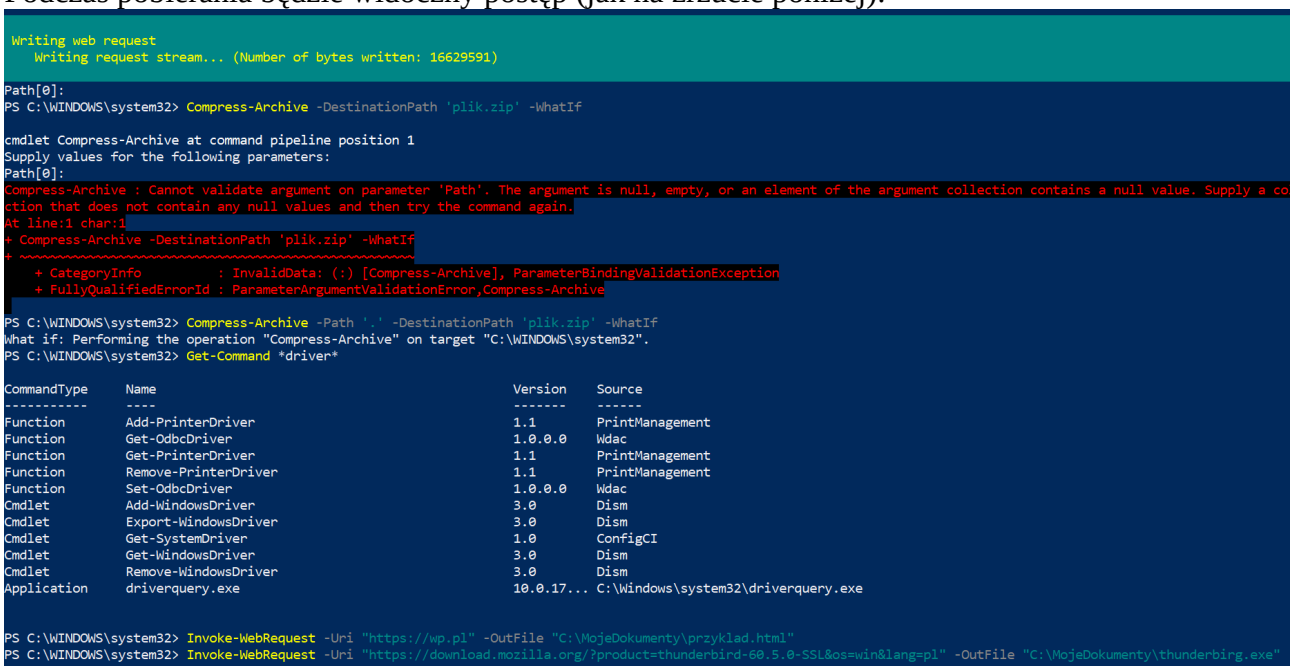
Invoke-WebRequest -Uri "https://wp.pl" -OutFile "C:\MojeDokumenty\przyklad.html"

Dzięki temu pobierzemy stronę główną (zawartość HTML) do wskazanego pliku

Kolejnym przykładem niech będzie pobranie pliku instalacyjnego Thunderbird:

Invoke-WebRequest -Uri "https://download.mozilla.org/?product=thunderbird-60.5.0-SSL&os=win&lang=pl" -OutFile "C:\MojeDokumenty\thunderbirg.exe"

Podczas pobierania będzie widoczny postęp (jak na zrzucie poniżej):



ZADANIA DO WYKONANIA:

UWAGA! Ze względu na sporą ingerencję w ustawienia systemowe (np. sieciowe) wszystkie zadania wykonujemy na maszynie WIRTUALNEJ! Przed przystąpieniem do jakichkolwiek modyfikacji najpierw należy spróbować odtworzyć wszystkie polecenia z materiału na własnej maszynie (celem przetestowania działania).

1. Czy da się utworzyć polecenie-skrypt, które będzie uruchamiało polecenia z uprawnieniami administratora użytkownikom, którzy nie posiadają takich uprawnień (ani nie należą do grupy administratorów)?
2. Czy da się utworzyć polecenie Start-Process, które będzie miało uruchomić inne polecenie Powershell (na kształt -BlockScript z Invoke-Command)?
3. Czym są pliki wim i esd? Jaka jest między nimi różnica?
4. Czym jest narzędzie Windows SIM (Windows ADK)? W jaki sposób można go użyć? Proszę wskazać jego możliwości na własnym przykładzie (użycie narzędzia).
5. Co oznaczają gwiazdki podawane w parametrach PowerShell?
6. W jaki sposób działa znak | w konsoli? W jaki sposób można go wykorzystać (najlepiej podać minimum dwa przykłady ilustrujące działanie wspomnianego przełącznika; dobrym przykładem może być np. polecenie Measure-Object lub Format-List/Format-Table)?
7. Czym różnią się od siebie pojęcia: dysk, partycja wolumen, vdisk? Należy podać przykłady zastosowania w diskpart/systemie Windows.

8. Dla maszyny wirtualnej należy utworzyć nowy wirtualny dysk o rozmiarze 20 GB (przydział dynamiczny). Następnie, poprzez poznane polecenia należy go zainicjować i podzielić na 5 partycji (po 4 GB każda). Przetestować możliwości podziału na MBR jak i GPT. Która wersja jest wygodniejsza do użycia. UWAGA – ćwiczenie ma zostać wykonane w linii poleceń, nie zaś w interfejsie graficznym (udokumentować zrzutami ekranu).
9. Partycje z poprzedniego polecenie połączyć w wirtualne przestrzenie danych (LVM). Co przez to zyskujemy?
10. Systemy z rodziny Microsoft Windows (Linux też) mogą, jeżeli dostawca internetowy tego nie pozwala, wykorzystywać sieć IPv6. Obsługę IPv6 (tzw. przejściową) umożliwia usługa Teredo. Poprzez wykorzystanie narzędzi konsolowych należy aktywować tę usługę i spróbować odpytać (ping) stronę o nazwie ipv6.google.com.
11. Dlaczego polecenia wydawane w kosnoli netsh mają niekiedy podane nazwy parametrów (np. name, gateway) a niekiedy pisane są ciągiem, bez podawania parametrów?
12. Należy znaleźć polecenia (netsh bądź PowerShell), które umożliwią zdobycie następujących informacji:
- hasła do zapisanej siec WLAN
- statystyk przesłanych danych na dowolnym interfejsie
- ustawienie na dowolnym kliencie nasłuchiwania DHCP (autokonfiguracja adresów i serwerów DNS) – przeciwne zachowanie do zaprezentowanego w dokumencie
- statystyki protokołu TCP (wraz z krótkim objaśniemiem na co wskazują)
13. W jaki sposób przekierować wyjście konsoli do pliku?
14. Czy jest możliwe pozyskanie klucza licencyjnego Windows poprzez PowerShell? Jeżeli tak to w jaki sposób można tego dokonać (bez wykorzystywanie aplikacji firm trzecich). Czy możliwe jest pobranie innych kluczy licencyjnych?
15. Czy można poprzez linię poleceń zmieniać nazwę komputera? Czy PowerShell pozwala na dołączenie komputera do Domeny/Grupy roboczej?
16. Czy istnieje możliwość poprzez PowerShell podmiany pulpitu Windows aktualnie zalogowanemu użytkownikowi? Jeżeli tak to jakie kroki należy podjąć by osiągnąć cel?
17. Czy da się przeszukiwać treść (np. pliku) poprzez polecenie PowerShell? Czy da się zwrócić ten wynik, odpowiednio sformatowany, do konsoli?

DODATKOWE (ocena celująca)

Przygotować wirtualny dysk przy użyciu konsoli. Następnie dysk ten podpiąć do menu startowego systemu Windows (bcdedit). Na dysku zainstalować dowolny system (Windows bądź Linux).

Informacje i materiały dodatkowe:

https://www.winhelponline.com/blog/run-command-elevated-administrator-ctrl-shift-enter-windows-10/

https://www.askvg.com/windows-10-secret-tip-open-programs-as-administrator-using-run-dialog-box/

https://stackoverflow.com/questions/32341507/how-open-powershell-as-administrator-from-the-run-window

https://superuser.com/questions/291687/is-it-possible-to-run-windows-without-gui/291694

https://www.petri.com/display-memory-usage-powershell

https://www.howtogeek.com/124087/how-to-create-a-shortcut-that-lets-a-standard-user-run-an-application-as-administrator/

https://social.technet.microsoft.com/Forums/en-US/132e170f-e3e8-4178-9454-e37bfccd39ea/startprocess-verb-runas-amp-credential?forum=winserverpowershell

https://docs.microsoft.com/en-us/windows-hardware/manufacture/desktop/enable-or-disable-windows-features-using-dism

https://www.neowin.net/news/guide-how-to-install-windows-10-using-vhds/

https://www.howtogeek.com/112660/how-to-change-your-ip-address-using-powershell/

https://stackoverflow.com/questions/7330187/how-to-find-the-windows-version-from-the-powershell-command-line