Dzień dobry Wszystkim Studentom,

1. Opis projektu zaliczeniowego:

Proszę stworzyć 4 maszyny wirtualne (WS2019, WS2016, Linux, W10)→ tworzymy środowisko systemów operacyjnych. Preferowany Hyper-V. Oczywiście można skorzystać z innego witrualizatora. Dla początkujących VirtualBox. Zasoby potrzebne do stworzenia poszczególnych maszyn wirtualnych:

System	HDD	RAM	Core
WS219	20 GB	2GB	1
WS2016	20 GB	2GB	1
W10	20 GB	2GB	1
Linux	15 GB	2GB	1
Unix	20 GB	2GB	1

Na Windows Server 2019 instalujemy i konfigurujemy usługę ADDS, DNS i DHCP → (zakres adresów w DHCP 10.0.0.1-10.0.0.15 w podsieci 255.255.255.0). Materiały znajdują się na OneDrive. Sieć komputerowa powinna być zbudowana tak, aby adresy statyczne serwerów były po kolei x.x.x.5 (WS2019), x.x.x.6 (WS2016). Pozostałe systemy operacyjne mają otrzymywać adresy IP automatycznie. Proszę ściągnąć system Solaris jako systemu unix. On również powinien działać w tej samej sieci 10.0.0.x. Każdy z państwa, ewentualnie Grup ćwiczeniowych→ tak dopuszczam takie rozwiązanie (ze względu na COVID-19 i prace zdalną omówimy to na zajęciach) Będzie musiał/musiała stworzyć w oparciu o firmę z dowolnej branży strukturę organizacyjną i wdrożyć ją w stworzone środowisko. Każde środowisko powinno zawierać co najmniej pięciu użytkowników i 5 działów, które zarządzane są poprzez Zasady grup (GPO). Zakres GPO pozostawiam Wam. Jeżeli ktoś nie będzie wiedział jak to sobie z tym tematem poradzić→ (studenci, którzy mieli ze mną zajęcia w poprzednim semestrze→robiliście to w projekcie zaliczeniowym.), proszę o kontakt lub pytanie na zajęciach.
Następnie przechodzimy do automatyzacji w pracy na naszych systemach operacyjnych. Po zbudowaniu w/w infrastruktury proszę o wykonanie skryptów, które będą automatycznie zmieniały ustawienia sieci, dodawały użytkowników do domeny z odpowiednimi uprawnieniami na dowolnym systemie operacyjnym, generowały raporty z eventlog z ostatnich 20 zdarzeń w systemie i zapisywały je do Katalogu c:\Raporty_System.

2. Dla osób, które chciały by podnieść ocenę.

Język skryptowy PowerShell proszę stworzyć:
Skrypt/skrypty które utworzą nam maszynę wirtualną (powinna znajdować się w tej samej sieci) Windows Server Core . Na tem serwerze zainstalować usługę IIS następnie utworzyć użytkownika ADMINWSBIIS (oczywiście skryptem) , który będzie mógł logować się również do pozostałych systemów operacyjnych w infrastrukturze. Skrypt/skrypty powinny pobierać informacje i przedstawiać je w formacie pliku html na powołanym wcześniej serwerze Windows Server Core z usługą IIS. Raport ma zawierać informacje jaki użytkownik wywołał skrypt, jakie jest obciążenie pamięci RAM podczas wykonywania skryptu/skryptów na dowolnym systemie w infrastrukturze. Aby dane były różne należy w na systemach wymusić działanie dowolnych programów i/lub usług aby dane nie były takie same. Zbierane informacje za pomocą wykonywanego skryptu, wykonywanych skryptów mają wyświetlać się w 10 różnych stronach, np.raport1.html, raport2.html,raport3.html,raport4.html itd w przeglądarce internetowej na serwerze Windows Serwer Core i być zapisywane na tej maszynie w katalogu c:\Raporty .
Każdy z plików ma zawierać informację na temat badanych elementów co 10 sekund kończąc na pliku raport10.html po czym zapytać nas czy chcemy jeszcze raz zacząć proces generowania nowych raportów poczynając od pliku raport1.html.

Z poziomu W10 raporty powinny być dostępne po wpisaniu do przeglądarki adresu "http://wscore/raport1.html", "http://wscore/raport2.html" etc.

OnDrive – aktualizowany):
https://tiny.pl/744r3
!^.a#}Y74We5yN]x?<h(

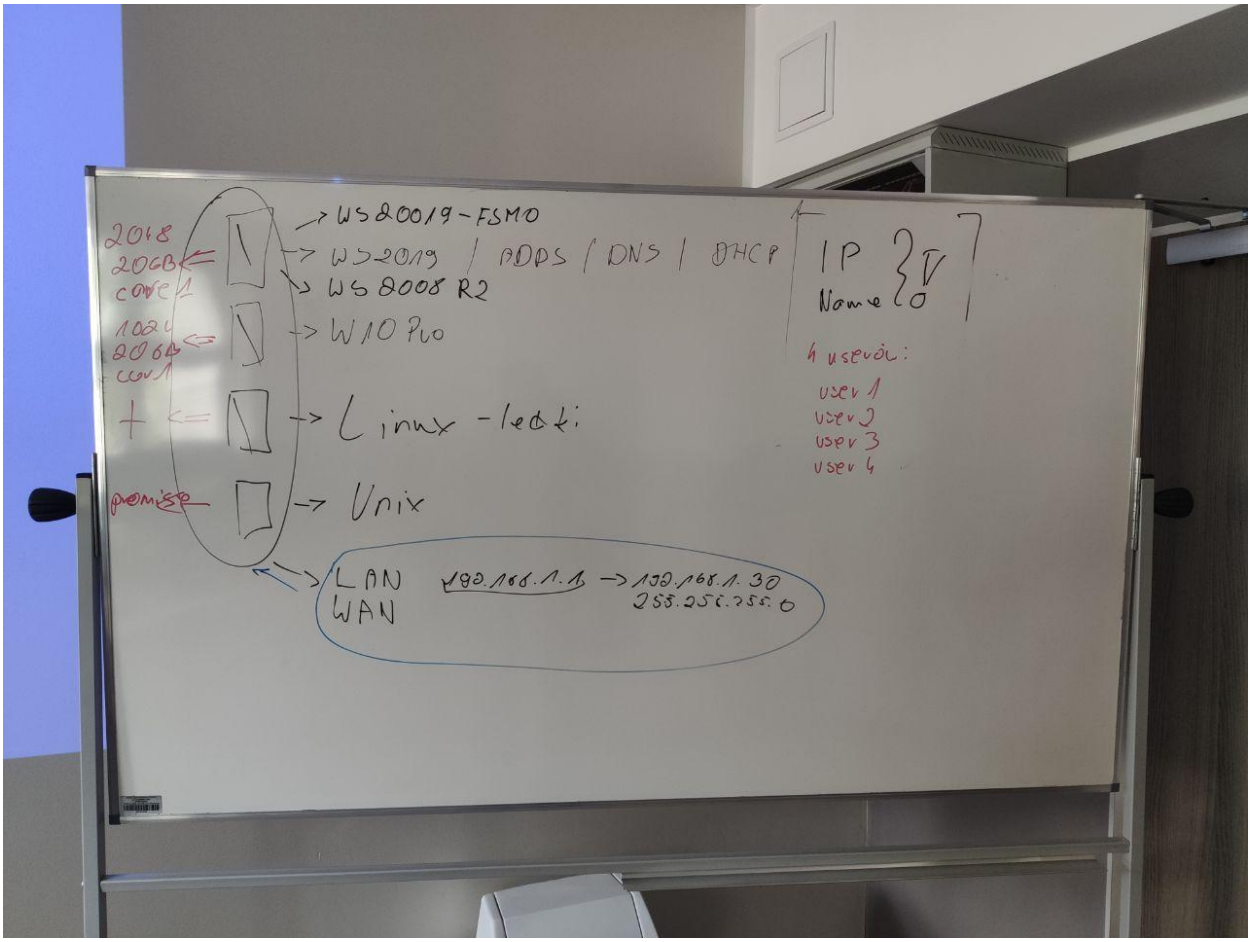
Pozostałe materiały zostaną udostępnione wkrótce w katalogu SO\Materiały dydaktyczne\ w OnDrive.

Materiały do pracy z Powershell.
Laboratoria do materiałów szkoleniowych znajdujących się na stronie:

https://web.microsoftstream.com/
user: student@s4gs4g.onmicrosoft.com
pass: Guv17228@

W razie pytań proszę o kontakt.

Materiał poglądowo-pomocniczny:



Laboratoria:

Praca z konsolą PowerShell - LAB

1. Uruchom konsolę znakową Powershell. (Jako administrator)
2. Uruchom rejestrowanie transkrypcji sesji do pliku c:\temp\1.txt
3. Wyświetl datę bieżącą
4. Wyświetl tylko samą datę (bez godziny)
5. Odgadnij polecenie wyświetlające informacje o dyskach
6. Odgadnij polecenie wyświetlające informacje o interfejsach sieciowych
7. Wyświetl zawartość pliku C:\Windows\System32\drivers\etc\hosts
8. Zatrzymaj transkrypt.
9. Korzystając z Show-Command przygotuj polecenie sprawdzające stan usługi Windows Update (wuauserv).
10. Przejrzyj parametry polecenia Get-Date. Korzystając z formatu yyyyMMdd wygeneruj datę w postaci RokMiesiacDzień (bez odstępów).
11. Sprawdź numer wersji Powershell.
12. Uruchom powershell.exe w trybie zgodności z wersją 2.0
13. Sprawdź czy wersja to rzeczywiście 2.0
14. Przygotuj ikonkę pozwalającą uruchamiać PowerShell zawsze „jako Administrator”

Integrated Script Editor - LAB

1. Uruchom ISE. Kolejne polecenia wpisuj w części pozwalającej na budowanie skryptu i uruchamiaj naciskając F8.

2. Korzystając z IntelliSense sprawdź jakie parametry ma polecenie Get-Content

3. Wybierz opcję -Encoding Sprawdź jakie wartości może przyjmować ta opcja. (BTW opcja przydaje się, jeżeli plik został utworzony z wykorzystaniem innego kodowania, niż obecnie używane w twojej sesji).
4. Skonstruuj komendę wyświetlającą tekst „Starting…" na czerwonym tle (skorzystaj z Write-Host i podpowiedzi IntelliSense).

5. Przekonaj się, że w ISE możesz mieć otwartych kilka niezależnych sesji PowerShell. Sprawdź działanie polecenia File >> New Powershell Tab

6. Zobacz działanie polecenia Edit >> Start Snippets

Aliasy komend - LAB

1. Korzystając z polecenia cd przejdź do katalogu głównego dysku C:
2. Sprawdź, do jakich komend odwołują się popularne aliasy:

a. Ls

b. Cat

c. Cd

d. Md

e. Cp

f. Mv

3. Utwórz alias o nazwie gd uruchamiający cmd-let Get-Disk
4. Sprawdź działanie tego aliasu
5. Wyświetl informacje o aliasie gd.
6. Usuń alias gd

Pomoc na temat komend - LAB

1. Zaktualizuj system pomocy

2. Wyświetl pomoc dla komendy Get-Process

3. Wyświetl pomoc dla komendy Get-Process w dedykowanym okienku

4. Wyświetl przykłady z helpa dotyczącego Get-Process

5. Wyświetl komendy dotyczące operacji wykonywanych na komputerze (*computer*)

6. Wyświetl dokłady opis komendy dotyczącej polecenia Checkpoint-Computer

7. Wyświetl komendy pozwalające wykonywać czynności na wykonywanie operacji na dyskach

8. Sprawdź jakie komendy znajdują się w module WindowsUpdate

System pomocy - LAB

1. Sprawdź ile „Parameter set-ów” ma polecenie Get-Content

2. Na czym polega różnica między nimi?

3. Czy polecenie Get-Content może wyświetlić zwartość kilku plików?

4. Sprawdź to! Utwórz 2 pliki w katalogu c:\temp. Wyświetl ich zawartość jedną komendą.

5. Czy parametr -Path jest konieczny? Sprawdź to!

6. Ile zakładek ma okno wyświetlane poleceniem Show-Command dla polecenia Get-Content?

7. Wyświetl pełny help na temat Get-Content

8. Pobierz zaktualizowany help do katalogu c:\temp

9. Wyświetl dostępne artykuły about

10. Odszukaj artykuł opowiadający o przekierowywaniu wyników komend. Jaką ma nazwę?

11. Wyświetl ten plik pomocy

Cz.2

Przekazywanie parametrów do cmd- letów - LAB

1. Wyświetl interfejsy sieciowe komputera, na którym pracujesz.

2. Spośród wszystkich interfejsów sieciowych wyświetl tylko te, w których nazwie wsytępuje „Ethernet"

3. Spośród wszystkich interfejsów sieciowych wyświetl tylko te, których nazwa zawiera „Ethernet" i które są interfejsami fizycznymi

4. Jak bardzo można skrócić nazwę parametru -Physical? Czy wystarczy -Phys? A może da się jeszcze krócej?

5. Polecenie Get-ItemProperty wyświetla więcej informacji o wybranym obiekcie. Wyświetl właściwości folderu C:\Windows

6. Wyświetl jedną komendą właściwości folderu C:\Windows i C:\Users

7. Wyświetl właściwości folderów C:\Windows i C:\Users zapamiętane uprzednio w zmiennej.

8. Utwórz plik, którego zwartością będą ścieżki z poprzednich przykładów

9. Wczytaj ścieżki z pliku do zmiennej i wywołaj dla nich Get-ItemProperty

Parametry WhatIf i Confirm - LAB

1. Sprawdź co by się stało gdyby uruchomić polecenie Restart-Computer?

2. Sprawdź co by się stało gdyby uruchomić polecenie Stop-Computer?

3. Wywołaj komendę wyłączającą komputer w taki sposób, aby przed wyłączeniem zostać zapytanym o potwierdzenie tej czynności. Tylko się nie pomyl, bo…. sam wiesz…

4. Wylistuj zawartość katalogu C:\Windows do pliku C:\temp\listing.txt

5. Wywołaj polecenie usuwająca plik listing.txt tak, aby zostać zapytanym o potwierdzenie usunięcia pliku. Na pytanie o usunięcie odpowiedz NIE.

6. Usuń ten plik w normalny sposób (bez dodatkowych potwierdzeń)

7. Jaki masz obecnie ConfirmPreference?

8. Utwórz jeszcze raz plik listing.txt

9. Zmień ConfirmPreference na Low

10. Uruchom polecenie kasujące plik bez dodatkowego parametru wymuszającego potwierdzenie. Czy było wymagane potwierdzenie? Na ewentualne pytanie odpowiedz NIE.

11. Zmień ConfirmPreference na Medium.

12. Uruchom polecenie kasujące plik bez dodatkowego parametru wymuszającego potwierdzenie. Czy było wymagane potwierdzenie? Na ewentualne pytanie odpowiedz NIE.

13. Otwórz nową sesję PowerShell. Sprawdź wartość ConfirmPreference w tej drugiej sesji. Czy wartość jest ustawiana globalnie czy per sesja?

14. Zmień ConfirmPreference na domyślną wartość.

15. Uruchom polecenie kasujące plik bez dodatkowego parametru wymuszającego potwierdzenie. Czy było wymagane potwierdzenie? Na ewentualne pytanie odpowiedz TAK.

Cz.3

Praca z potokami - LAB

1. Odnajdź polecenia, które mają coś do czynienia z drukowaniem

2. Wyświetl zainstalowane drukarki

3. Wyświetl informacje tylko o jednej wybranej drukarce posługując się jej nazwą. Drukarkę identyfikujesz po nazwie. Dalej zawsze używaj tej jednej drukarki.

4. Upewnij się, że drukarka jest wyłączona. Tak naprawdę nie chcemy tutaj niczego drukować ;) Wydrukuj kilka plików na jedną z drukarek.

5. Wyświetl zadania drukowania na wybranej drukarce

6. Umieść informacje uzyskane w poprzednim przykładzie w pliku c:\temp\printjobs.txt

7. Przejrzyj listę zadań drukowania w przeglądarce graficznej

8. Zobacz jakie właściwości ma zadanie drukowania

9. Zobacz jakie właściwości ma drukarka

10. Wybierz jedno zadanie drukowania. Jeśli nie pamiętasz identyfikatorów zadań wykonaj jeszcze raz zadanie 5. Wyświetl dokładne informacje tylko o tym jednym zadaniu

11. Usuń to jedno wybrane zadanie z kolejki drukowania. Wyświetl ponownie listę zadań drukowania i upewnij się, że to zadanie zniknęło

12. Usuń wszystkie zadania drukowania.. Upewnij się, że kolejka jest pusta (ewentualnie jedno zadanie jest w statusie „Deleting”, po chwili i to zadanie powinno zniknąć)

Sort-Object - LAB

1. Wyświetl listę procesów

2. Posortuj listę procesów wg ilości czasu procesora (właściwość CPU)

3. Posortuj ją w kolejności CPU malejąco

4. Czy istnieją procesy o tej samej nazwie ale z różną intensywnością wykorzystujące procesor? Sprawdź to sortując listę procesów wg nazwy i CPU

5. Jaka jest pełna nazwa właściwości do której odwołuje się CPU?

6. Napisz polecenie z punktu (3) które odwołuje się do pełnej nazwy właściwości.

Measure-Object - LAB

1. Wyświetl listę zainstalowanych HotFix

2. Sprawdź ile jest zainstalowanych poprawek

3. Wyświetl wszystkie certyfikaty z folderu cert:\LocalMachine wraz z podkatalogami

4. Policz ile certyfikatów jest zainstalowanych

5. Jaką datę ma certyfikat, który najwcześniej wygaśnie (zwróć uwagę na włsściwość NotAfter)

6. Jaki jest najmniejszy i największy plik z katalogu c:\windows\system

Select-Object -LAB

1. Polecenie Get-Volume wyświetla informacje o wolumenach, sprawdź jego działanie

2. Zobacz jakie właściwości i metody posiadają zwracane przez Get-Volume obiekty

3. Wyświetl tylko następujące informacje o wolumenach: litera dysku, rozmiar dysku, pozostałe wolne miejsce na dysku

4. Posortuj wynik tak, aby dyski z największym wolnym obszarem były na pierwszym miejscu

5. Wyświetl tylko jeden dysk z największym wolnym obszarem

6. Wyświetl listę procesów

7. Posortuj wynik wg CPU malejąco

8. Wybierz tylko 5 procesów najbardziej intensywnie wykorzystujących CPU

9. Wyświetl 1000 ostatnich zdarzeń z dziennika zdarzeń „Application”

10. Wyświetl tylko właściwość Source dla tych ostatnich 1000 zdarzeń

11. Wyświetl tylko unikalne „source” występujące wśród ostatniego 1000 zdarzeń.

Select i kolumny wyliczane - LAB

1. Wyświetl certyfikaty z magazynu Local Machine >> CA (cert:\LocalMachine\CA)

2. Wyświetl tylko właściwości: Thumbprint, NotBefore i NotAfter (to są daty ważności certyfikatu).

3. Wyświetl dodatkową kolumnę wyliczaną jako różnicę w dniach między NotAfter i NotBefore (na rzecz $_.NotAfter wywołaj metodę Subtract podając jako argument $_.NotBefore.

4. Wyświetl listę procesów

5. Posortuj listę procesów wg CPU malejąco

6. Wartość CPU jest podawana w sekundach. Przelicz wartość na minuty. Wyświetl tylko 5 najbardziej obciążających CPU procesów

7. Zmień formatowanie wyniku tak, aby czas był wyświetlany w minutach (bez miejsc po przecinku)

Eksport i import - LAB

1. Zapisz listę plików z katalogu c:\windows w postaci pliku HTML

2. Twoim zadaniem jest raz na tydzień zapisać w pliku informację o wolnym i zajętym miejscu na dysku. Przygotuj komendę, która wyświetli literę dysku, rozmiar dysku i rozmiar wolnego obszaru na dysku dla dysku C:

3. Wyeksportuj wynik tej komendy do pliku report.csv w formacie CSV. Zadbaj o to aby koljne uruchomienia polecenia dopisywały kolejny rekord do pliku.

4. Zaimportuj wynik z pliku i wyświetl go na ekranie

5. Dodaj do polecenia z punktu 2 wyrażenie, które spowoduje, że do wyświetlanego wyniku zostanie dołączona informacja o bieżącej dacie

6. Wyeksportuj wynik z poprzedniego polecenia do pliku report_data.csv

7. Zaimportuj wyniki z pliku

8. Zaimportuj wyniki z pliku, ale wyświetl tylko ostatni zapis.

Where-Object - LAB

1. Wyświetl listę plików z katalogu c:\temp starszych niż 5 minut. Aby wyznaczyć czas „5 minut temu” użyj (Get-Date).AddMinutes(-5)

2. Wyświetl listę plików z katalogu c:\temp starszych niż 2 dni i mających rozszerzenie .txt. Aby wyznaczyć czas „2 dni temu” użyj (Get-Date).AddDays(-2)

3. Uruchom kilka procesów notatnika

4. Wyświetl listę procesów

5. Wyświetl listę procesów i wyfiltruj z nich tylko te, które mają nazwę procesu „*notepad*”

6. Wyświetl zdarzenia z dziennika zdarzeń system, które pochodzą z source „USER32”

7. Wyfiltruj spośród tych zdarzeń te, które mają EventId równy 1074. Zobaczysz informacje o wykonywanych włączeniach komputera.

}

Foreach-Object - LAB

1. Utwórz plik „services.txt” o zawartości wuauserv bits

2. Wczytaj zawartość pliku.

3. Połącz polecenie z poprzedniego punktu potokiem z poleceniem zatrzymującą usługi o nazwach takich jak wpisane w pliku. (Pamiętaj o uruchomieniu powershella jako administrator)

4. Zmień polecenie z poprzedniego punktu tak, aby te usługi były uruchamiane.

5. Dodaj do poprzedniego polecenia dodatkową instrukcję powodującą wyświetlenie napisu „Starting …”. W miejscu kropek ma się pojawić nazwa startowanej usługi. (Kiedy w jednym bloku chcesz uruchamiać kilka komend rozdziel je znakiem średnika).

6. Dodaj blok poleceń wykonywany przed uruchamianiem usług, który wyświetli komunikat „STARTING SERVICES”. (Jeśli komenda jest zbyt długa, to możesz zakończyć linijkę znakiem ` - odwrócony apostrof – na klawiaturze na tym samym klawiszu co tylda ~. Dla Powershella to informacja, że ciąg dalszy komendy znajduje się poniżej.)

7. Dodaj blok poleceń uruchamianych po uruchomieniu lub zatrzymaniu usług, który wyświetli komunikat „DONE”.

Dlaczego działa potok? - LAB

1. Wyświetl dostępne wolumeny dyskowe

2. Skonstruuj polecenie, które wyświetli informacje tylko o wtych wolumenach dyskowych, które są przekazywane potokiem rozpoczynającym się od: ‘C’ | …. (jeśli na tym etapie otrzymujesz błąd – kontynuuj do następnego kroku)

3. Wyświetl help dotyczący Get-Volume. Zwróć uwagę na to, jak można przekazywać potokiem wartość dla parametru DriveLetter

4. Skonstruuj polecenie, który zmieni nazwę kolumny na „DriveLetter” dla potoku rozpoczynającego się od ‘C’ | ….

5. Korzystając z potoku z poprzedniego zadania połącz go z poleceniem Get-Volume, tak aby wyświetlane były tylko informacje o wolumenach przekazywanych potokiem.

6. Jak wykonać to samo zadanie z wykorzystaniem ForEach?

Cz.4

Formatowanie wyników - LAB

1. Polecenie
   `Get-WMIObject -Class Win32_Desktop`
   wyświetla informacje o ustawieniach pulpitu dla użytkowników komputera

2. Wyświetl wynik w/w polecenia w postaci skróconego zestawienia (tylko same nazwy użytkowników)

3. Wyświetl wynik w postaci 3 kolumn

4. Sformatuj wynik do postaci tabeli

5. Wyświetl wynik polecenia w postaci listy

6. Wyświetl infromacje w postaci tabeli, ale na ekranie mają się pokazać tylko Name i informacje o konfiguracji wygaszacza ekranu

7. Wyświetl wyniki polecenia
   `Get-WmiObject -Class win32_desktop`
   w przeglądarce graficznej (GridView)

Cz. 5

Praca z obiektami WMI i CIM - LAB

1. Wylistuj obiekty WMI, które w nazwie zawierają *network*. Odszukaj klas, które mogą mieć coś wspólnego z konfiguracją kart sieciowych

2. Wykonaj tą samą czynność dla klas CIM

3. Korzystając z metod WMI i CIM: Wyświetl obiekty klasy WIN32_NetworkAdapter

4. Korzystając z metod WMI i CIM: Sprawdź jakie właściwości i metody ma obiekt klasy Win32_NetworkAdapter

5. Korzystając z metod WMI i CIM: Wyświetl dla każdego obiektu z poprzednich punktów tylko właściwość Caption oraz MACAddress.

6. Korzystając z metod WMI i CIM: Zbadaj, czy komputer na jakim pracujesz jest wirtualny. W tym celu wyświetl klasę Win32_BIOS

7. Korzystając z metod WMI i CIM: Korzystając z klasy Win32_UserAccount wyświetl listę lokalnych użytkowników komputera

WMI i CIM. Praca z obiektami właściwościami i metodami - LAB

1. Korzystając z klasy Win32_NetworkAdapter wylistuj wszystkie karty sieciowe.

2. Sprawdź jakie metody i właściwości posiadają obiekty klasy Win32_NetworkAdapter.

3. Przygotuj komendę zwracające jedną wybrany przez ciebie interfejs sieciowy

4. Korzystając z metody Disable wyłącz ten adapter

5. Korzystając z metody Enable włącz ten adapter

6. Korzystając z klasy Win32_NetworkAdapterConfiguration wylistuj te adaptery sieciowe, które mają włączone DHCP.

7. Sprawdź jakie metody i właściwości posiadają obiekty klasy Win32_NetworkAdapterConfiguration

8. Korzystając z metody ReleaseDHCPLease zwolnij adres IP uzyskany z serwera DHCP dla wybranego przez ciebie interfejsu sieciowego

9. Korzystając z metody RenewDHCPLease pobierz adres IP z serwera DHCP ponownie.

Sesje CIM - LAB

1. Korzystając ze zmiennej $env:COMPUTERNAME sprawdź nazwę komputera, na którym obecnie pracujesz. Tej nazwy będziemy używać w kolejnych zadaniach. Jeżeli masz do dyspozycji więcej komputerów, możesz wykonywać kolejne polecenia względem innego komputera. Jeśli nie, to sesje będziemy nawiązywać z komputerem lokalnym używając przed chwilą ustalonej nazwy.

2. Korzystając z obiektu sesji podłącz się do komputera zdalnego i sprawdź jaki na nim jest zainstalowany system operacyjny. (Skorzystaj z klasy Win32_OperatingSystem)

3. Korzystając z obiektu sesji podłącz się do komputera zdalnego i sprawdź jakie aplikacje są na nim zainstalowane. (Skorzystaj z klasy Win32_Product)

4. Zauważ dodatkową kolumnę PSComputerName informującą o tym z jakiego komputera dane są pobierane.

5. Wyświetl aktualnie dostępne sesje

6. Usuń wszystkie sesje

7. Utwórz obiekt sesji oparty o protokół DCOM

8. Utwórz obiekt sesji do wybranego komputera z wykorzystaniem przed chwilą utworzonego obiektu sesji.

9. Korzystając z obiektu sesji wyświetl informacje o drukarkach dostępnych na zdalnym komputerze.

10. Usuń wszystkie obiekty sesji

Cz. 6

Powershell - zmienne - LAB

1. Zadeklaruj zmienną o nazwie MyService i przypisz do niej wartość „bits”

2. Wyświetl usługi o nazwie zgodnej z nazwą wpisaną do zmiennej MyService

3. Korzystając z poleceń dedykowanych do obsługi zmiennych zadeklaruj zmienne

a. EventLogName i przypisz jej wartość System

b. EventNumber i przypisz jej wartość 5

4. Zmień polecenie wyświetlające 3 ostatnie zdarzenia z aplikacyjnego dziennika zdarzeń, tak aby korzystało z wcześniej zdefiniowanych zmiennych:

Get-EventLog -Newest 3 -LogName application

5. Wyświetl zawartość napędu ENV: (Ten napęd zawiera zmienne środowiskowe zdefiniowane na danym komputerze. Można ich używać w swoich skryptach.)

6. Nie wykorzystując instrukcji dedykowanych do pracy ze zmiennymi (w uproszczony sposób – notacja z $) utwórz zmienne i przypisz im odpowiednie wartości:

a. MyComputerName

b. MyOperatingSystem

c. MyUserName

Uwaga: Zmienne środowiskowe z napędu ENV: są w powershell traktowane jak pliki. Aby wczytać ich wartość posłuż się poleceniem Get-Content, np. tak:

Get-Content -Path Env:\PROCESSOR_ARCHITECTURE

7. Do zmiennych środowiskowych można się również odwoływać przez zmienną $env. Pozwala to uprościć konstrukcje z poprzedniego zadania. Odwołaj się do tych samych zmiennych środowiskowych korzystając z notacji podobnej do:

$env:PROCESSOR_ARCHITECTURE

Zmienne tablicowe - LAB

1. W katalogu c:\temp trzeba będzie utworzyć pewien zestaw podkatalogów. Nazwy tych katalogów będą docelowo znajdować się w pliku c:\temp\subdirs.txt. Na razie jednak zacznij od zadeklarowania tablicy $subdirs, której elementami będą napisy:

a. 01_Input

b. 02_Processing

c. 03_Results

2. Zadeklaruj zmienną BaseDir i zainicjuj ją wartością ‘c:\temp\’.

3. Przekaż zmienną $subdirs potokiem do polecenia foreach-object, które na razie tylko wyświetli zawartość poszczególnych elementów tablicy.

4. Zmień polecenie w ForEach-Object tak aby wyświetlało całą ścieżkę powstałą z połaczenia zmiennej BaseDir z w danej chwili przetwarzanym podkatalogiem

5. Zmień polecenie w ForEach-Object z wyświetlającego bieżący element na tworzące katalog

6. Korzystając z potoku i polecenia Out-File umieść zawartość zmiennej $subdirs w pliku c:\temp\subdirs.txt.

7. Wyświetl zawartość tego pliku

8. Zmień polecenie z punktu (4) tak, aby nazwy podkatalogów wczytywać z pliku, a nie korzystać ze zmiennej $subdirs

9. Korzystając z notatnika zmień nazwy katalogów zapisane w pliku c:\temp\subdirs.txt na:

a. A_Input

b. B_Processing

c. C_Results

d. D_Temporary

10. Zmień polecenie z punktu (8), tak aby tworzyło podkatalogi (podobnie jak to robiłeś w pkt. (5)). Uruchom polecenie.

Typy zmiennych - LAB

1. Twoim zdaniem jest utworzenie zmiennej, która może dalej posłużyć jako nazwa pliku identyfikujaca kiedy plik powstał. Zadeklaruj zmienną napisową typu string o nazwie LogFile.

2. Korzystając z literałów YYYY_MM_DD oraz metody ToString typu DateTime wyświetl aktualna datę w postaci odpowiednio sformatowanego napisu

3. Zainicjuj zmienną $LogFile wartością bieżącej daty – czyli połącz to co zostało zrobione w punkcie (1) oraz (2)

4. Zmodyfikuj odpowiednio polecenie z poprzedniego punktu tak aby oprócz daty zmienna zawierała też godzinę.

5. Twoim zadaniem jest zmierzenie czasu, przez który będzie pracować pewna aplikacja (u nas notepad). Zadeklaruj zmienne typu Datetime:

a. $StartTime

b. $StopTime

6. Przypisz do zmiennej $StartTime bieżący czas

7. Korzystając z polecenia poniżej uruchom notatnik:

Start-Process -FilePath "notepad.exe" -Wait

(Komenda uruchamia notatnik, ale nie kończy sie od razu, tylko czeka na zakończenie uruchomionego procesu.

8. Przypisz do Zmiennej $StopTime bieżączy czas.

9. Zadeklaruj zmienną $TimeSpent typu TimeSpan i przypisz jej różnicę między StopTime a $StartTime. Użyj metody Subtract wywoływanej na rzecz $StopTime, która wyznaczy poszukiwaną różnicę

10. Wyświetl wyznaczoną różnicę w sekundach.

Cz. 7

Obsługa błędów w PowerShell - LAB

1. Aby obsłużyć błąd najpierw sprowokujemy jego wystąpienie. W tym celu najpierw wyłączymy usługę bits. Do zmiennej bits pobierz za pomocą Get-WmiObject usługę bits.

2. Sprawdź aktualny status usługi (właściwość StartMode)

3. Korzystając z metody ChangeStartMode zmień tryb uruchomienia na Disabled

4. Zatrzymaj usługi bits i winrm

5. Sprawdź czy usługi są rzeczywiście zatrzymane.

6. Sprawdź wartość zmiennej $ErrorActionPreference (powinna wynosić Continue)

7. Wpisz polecenia uruchamiające bits i winrm i uruchom je jednocześnie. Przy uruchamianiu usługi bits powinien pojawić się błąd.

8. Sprawdź status usług bits i winrm. Czy winrm działa? Dlaczego?

9. Zmień wartość globalnej zmiennej ErrorActionPreference na Stop.

10. Zatrzymaj usługi bits i winrm

11. Sprawdź czy usługi są rzeczywiście zatrzymane.

12. Wpisz polecenia uruchamiające bits i winrm i uruchom je jednocześnie. Przy uruchamianiu usługi bits powinien pojawić się błąd.

13. Sprawdź status usług bits i winrm. Czy winrm działa? Dlaczego?

14. Zmień polecenie uruchamiające usługi tak aby niezależnie od wartości globalnej zmiennej ErrorActionPreference, jeśli dochodzi do błędu kolejne polecenia były wykonywane bez zgłaszania komunikatów o błędzie.

15. Sprawdź status usług bits i winrm. Czy winrm działa? Dlaczego?

16. Przywróć początkowe ustawienia trybu uruchomienia usługi bits.

Instrukcja try/catch - LAB

1. Utwórz na lokalnym dysku plik tekstowy, np. FileToCopy.txt w katalogu c:\temp.

2. Przygotuj i uruchom polecenie kopiujące ten plik na zasób sieciowy, np. \\server01\C$\temp\NewFile.txt Po uruchomieniu polecenie zakończy się błędem

3. Korzystając z instrukcji try catch zmień polecenie tak, aby w przypadku błędu wyświetlało komunikat „file cannot be copied”

4. Zmień polecenie wyświetlające komunikat o błędzie tak aby wskazać na przyczynę błędu (skorzystaj ze zmiennej $Error[0].Exception.Message)

5. Dodaj za instrukcją kopiującą plik polecenie wyświetlające komunikat „file copied”

6. Zmień polecenie kopiujące plik tak aby nie było błędu (zmień np. ścieżkę docelową na poprawną). Uruchom polecenie w aktualnej postaci i upewnij się, że komunikat dodany w kroku (5) został wyświetlony.

Cz. 8

Instrukcja IF - LAB

1. Twoim zadaniem będzie przekopiować plik między dwoma komputerami. Aby uniknąć błędu w pierwszej kolejności sprawdzisz czy zdalny komputer odpowiada na ping. Wykonaj testowe sprawdzenie łączności ze zdalnym komputerem za pomocą polecenia Test-Connection. Postaraj się aby polecenie:

a. Zwracało wynik prawda/falsz

b. W przypadku niedostępności zdalnego komputera na ekranie nie pojawiał się komunikat o błędzie

c. Wykonywało test tylko w oparciu o odpowiedź na pojedynczy pakiet ping

2. Wynik polecenia Test-Connection zapamiętaj w zmiennej $isAlive

3. Wykorzystaj zmienną $isAlive w warunku polecenia if. Jeśli warunek jest spełniony plik ma być kopiowany z komputera lokalnego na zdalny

4. Dodaj do polecenia wyrażenie else. W przypadku braku łączności z komputerem zdalnym wyświetl komunikat „remote host is not responding”

5. Napisz polecenie, które uruchomi usługę tylko o ile aktualnie ta usługa nie jest uruchomiona. Przed i po uruchomieniu usługi dodaj polecenia wyświetlające na ekranie dodatkowe komunikaty. Dodaj do poprzedniej instrukcji polecenie else, które w przypadku gdy usługa już działa wyświetli komunikat „Service is already running”

Instrukcja SWITCH - LAB

1. Poniższe polecenie wyświetla informacje o statusie zainstalowanego oprogramowania na komputerze, w tym informacje o stanie aktywacji licencji: Get-WmiObject -Class SoftwareLicensingProduct -Filter "Name LIKE '%Windows%' AND PartialProductKey <> null"

2. Upewnij się że polecenie zwraca rekord odpowiedzialny za system windows i zapisz go w zmiennej $license.

3. Sprawdź zawartość właściwości LicenseStatus utworzonej przed chwilą zmiennej.

4. Liczba znajdująca się w tej właściwości oznacza status aktywacji systemu windows. Mając na uwadze wartości umieszczone poniżej przygotuj konstrukcję switch, która wyświetli tekstowy ekwiwalent dla wartości liczbowej w $license.LicenseStatus:

a. 0 "Unlicensed"

b. 1 "Licensed"

c. 2 "OOBGrace"

d. 3 "OOTGrace"

e. 4 "NonGenuineGrace"

f. 5 "Notification"

g. 6 "ExtendedGrace"

5. Zapisz wynik polecenia w zmiennej $licenseDescription.

6. Korzystając ze zmiennej wyświetl komunikat w postaci The system license status is ...

Pętla WHILE - LAB

1. W tym zadaniu należy przygotować zestaw plików o nazwach 1.txt, 2.txt itd. W efekcie powstanie skrypt, który zrealizuje to zadanie. Kolejne punkty realizowane krok po kroku pokierują cię jak to zrobić.

2. Zadeklaruj zmienne $i i $max. Zainicjuj je wartościami 0 i 30

3. Zadeklaruj zmienną $sourceFileName i zainicjuj ją wartością ‘c:\temp\master.txt’

4. Umieść w pliku wskazywanym przez $sourceFileName napis „This is a master configuration file”

5. Zadeklaruj zmienną $destinationFolder i zainicjuj ją wartością ‘c:\temp\distribution’

6. Skonstruuj wyrażenie IF, które sprawdza istnienie katalogu wskazywanego przez $destinationFolder i jeśli nie ma takiego folderu to go tworzy. (Skorzystaj z funkcji Test-Path)

7. Przygotuj pętlę WHILE, która będzie wykonywana tak długo póki $i jest mniejsze od $max

8. W pętli dodaj instrukcję zwiększającą $i o 1.

9. Zadeklaruj zmienną $destinationFile i zainicjuj ją wartością w postaci $i.txt

10. Korzystając z polecenia Join-Path połącz nazwę $destinationFolder z $destinationFile i zapamiętaj wynik w zmiennej $newFileName

11. Skopiuj plik $sourceFile na $newFileName i wyświetl komunikat „File … has been created” Przetestuj działanie poleceń

}

Pętla FOR - LAB

1. Utwórz kilka plików w wybranym katalogu (tutaj c:\temp)

2. Zadeklaruj zmienną $files i przypisz do niej wynik polecenia Get-ChildItem listującego zawartość folderu c:\temp

3. Napisz pętlę foreach,która dla każdego pliku z kolekcji $files:

a. Zaszyfruj ten plik (wywołaj metodę Encrypt() dla bieżącego elementu kolekcji)

b. Zmień atrybut ReadOnly na $true. Wykorzystaj w tym celu właściwość ReadOnly bieżącego elementu kolekcji

4. Sprawdź czy pliki są zaszyfrowane i mają ustawiony atrybut tylko do odczytu (skorzystaj np. z eksploratora windows)

5. Napisz koleją pętlę foreach, która:

a. Odszyfruje pliki

b. Wygasi atrybut tylko do odczytu

6. Polecenia

a. [console]::beep(500,300) – powoduje wygenerowanie dźwięku (500 Hz, 300 ms)

b. Start-Sleep – zatrzymuje działanie skryptu na określony czas Napisz pętlę for, która wygeneruje 5 dźwięków w odstępach jednosekundowych

cz. 9

Ustawienia bezpieczeństwa skryptów
- LAB

1. Utwórz skrypt zawierający następujące polecenie:

Get-WMIObject win32_Processor | Select CurrentClockSpeed

Zapisz go na dysku jako c:\temp\MyScript.ps1

2. Sprawdź Execution Policy.

3. Zmień Execution Policy na AllSigned

4. Spróbuj uruchomić skrypt korzystając ze ścieżki bezwzględnej i względnej

5. Uruchom powershell.exe przekazując do niego parametr pozwalający na uruchamianie lokalnych niepodpisanych skryptów

6. Uruchom w nowej powłoce swój skrypt

7. Zamknij powłokę otwartą w punkcie (5)

8. Zmień ExecutionPolicy na RemoteSigned

9. Sprawdź tak jak w pkt 4, czy skrypt teraz się uruchamia

10. [Powtórz kroki zaprezentowane w 5’ filmu w celu wygenerowania certyfikatu do podpisywania kodu i podpisz nim swój skrypt, a następnie sprawdź jego działanie.]

Skrypty, funkcje, moduły - LAB

1. Do katalogu c:\temp wgraj kilka plików o różnych datach modyfikacji. Dalszą cześć zadań dopasuj do tych dat. Celem mogło by być przygotowanie skryptu, który będzie z określonego folderu przenosił lub usuwał pliki starsze niż określona data. Aby zadanie miało charakter powtarzalny, my zdecydujemy się tylko wyświetlać te pliki.

2. Przygotuj polecenie, które wyświetli pliki starsze niż 2016-09-16 (dopasuj datę tak, aby tylko część plików była wyświetlana).

3. Zamień stałe odpowiadające za ścieżkę katalogu oraz datę na zmienne, które należy zadeklarować na początku tworzonego skryptu. Popraw deklaracje tak, aby zmienne miały jawnie określony typ i przypisz im wartości.

4. Zastąp stałe używane w poleceniu z punktu (2) na zmienne zadeklarowane w punkcie (3).

5. Zmień deklaracje zmiennych z punktu (3) na definicje parametrów skryptu. Pamiętaj o zastosowaniu modyfikatorów jak: cmdbinding, mandatory oraz określeniu wartości domyślnych tam gdzie ma to sens. Zapisz skrypt w pliku o nazwie FilesOlderThan.ps1

6. Sprawdź działanie przygotowanego skryptu.

7. Zmień swój skrypt w funkcję Get-FilesOlderThan i zapisz w pliku o nazwie utilities.psm1

8. W nowym oknie powershell załaduj moduł do pamięci i uruchom funkcję

Moduły powershell - LAB

1. Wyświetl dostępne na systemie napędy PowerShell (PSDrive)

2. Utwórz katalog C:\PowershellLib i zapisz w nim właśnie tworzony skrypt

3. Utwórz nowy napęd wirtualny PowerShell o nazwie Lib wskazujący na utworzony przed chwilą folder

4. Wylistuj zawartość tego katalogu

5. Wyświetl listę załadowanych modułów

6. Wyświetl listę dostępnych modułów

7. Wyświetl listę poleceń znajdujących się w module NetTCPIP

8. Sprawdź działanie poleceń Get-NetIPAddress

9. Sprawdź listę załadowanych modułów. Powinien się na niej znaleźć także NetTCPIP, który został załadowany automatycznie, bez użycia Import-Module, podczas uruchamiania polecenia z tego modułu.

10. Usuń moduł z pamięci i zweryfikuj, że moduł został poprawnie wyładowany z pamięci

11. Przejdź do rejestru do HKEY_LOCAL_MACHINE:\Software

12. Utwórz nowy klucz rejestru o nazwie TestSoft

13. Dodaj dwie wartości do klucza rejestru: Edition – typu string – o wartości ‘Professional’ i Version – typu DWord – o wartości 10

14. Wyświetl wszystkie właściwości w kluczu TestSoft

15. Wyświetl wartość właściwości Edition

16. Zmień wartość właściwości Edition na ‘Enterprise’

17. Usuń właściwość Version Usuń klucz TestSoft i wróć na dysk C:

Cz. 10

Remoting - uruchamianie poleceń zdalnych - LAB

1. Włącz remoting na lokalnym komputerze korzystając z opcji SkipNetworkProfileCheck, aby pominąć krok sprawdzania czy wśród połączeń sieciowych znajdują się takie które są sklasyfikowane jako publiczne.

2. Nawiąż połączenie do komputera lokalnego za pomocą polecenia Enter-PSSession

3. Wyświetl nazwę komputera, do którego połączenie zostało nawiązane

4. Zakończ połączenie

5. Uruchom na systemie zdalnym polecenie wyświetlające nazwę komputera w sposób wsadowy (nieinteraktywny) korzystając z polecenia Invoke-Command

6. Wyświetl wszystkie właściwości zwrócone przez zdalnie uruchomione polecenia

7. Zwróć uwagę, że wśród właściwości znajduje się PSComputerName, które zawiera informacje o komputerze na którym polecenie było wykonywane. Wyświetl PSComputerName.

Uwierzytelnianie sesji remoting - LAB

1. Wyświetl okno pytające o dane uwierzytelnienia bez żadnych dodatkowych parametrów

2. Wyświetl okno pytające o dane uwierzytelnienia. Niech komunikat wyświetlany w oknie będzie następujący: „Enter username and password from MYDOMAIN"

3. Wyświetl okno pytające o dane uwierzytelnienia:

a. Komunikat w oknie – „Enter Password"

b. Wstępnie wypełniona nazwa użytkownika – w postaci <nazwa komputera lokalnego>\Administrator

c. Wartość zwracana ma być zapisana do zmiennej $adminCred

4. Wyświetl opcje klienta protokołu WSMAN

5. Wyświetl wartość opcji TrustedHosts korzystając z polecenia Get-Item

6. Zmień wartość opcji TrustedHosts na *

7. Wyświetl wartość opcji TrustedHosts

8. Zapisz do zmiennej $currentValue aktualną wartość opcji TrustedHosts. Sprawdź zawartość tej zmiennej

9. Korzystając z danych uwierzytelnienia pobranych w kroku (3) uruchom na lokalnym komputerze skrypt wyświetlający „Current user is $($env:USERNAME) Uwaga! Jeśli konto administratora na bieżącym komputerze jest zablokowane (wyłączone) to skorzystaj z innego konta.

Przekazywanie argumentów - LAB

1. Polecenie *Get-ItemProperty -Path HKLM:\SOFTWARE\Wow6432Node\Microsoft\Windows\CurrentVersion\Uninstall* | Where { $_.DisplayName -ne $null }* wyświetla listę zainstalowanych na komputerze aplikacji. Sprawdź działanie tego polecenia

2. Aby wyświetlić tylko wybrane informacje z tej listy dodaj w potoku polecenie, które pokaże jedynie: DisplayName, Publisher, DisplayVersion, InstallDate

3. Aby wyświetlić zainstalowane oprogramowanie w bardziej czytelnej postaci dodaj w potoku polecenie sortowania w kolejności alfabetycznej wg DisplayName

4. Zadeklaruj zmienne

a. $DisplayName – zainicjuj ją wartością „C++" (w przypadku braku na komputerze aplikacji zawierającej w nazwie C++ zmień tę wartość na inną)

b. $Publisher – zainicjuj ją wartością „Microsoft"

5. Aby wyfiltrować tylko określone oprogramowanie dodaj do tworzonej komendy w potoku polecenie, które wyświetli tylko programy zawierające w nazwie $DisplayName i wydane przez dostawców, którzy w nazwie mają ciąg znaków znajdujący się w zmiennej $Publisher

6. Zmień polecenie tak, aby z wykorzystaniem remotingu można je było uruchomić na systemach zdalnych. Pamiętaj o poprawnym przekazaniu parametrów do polecenia zdalnego. Jako przykładowego komputera zdalnego użyj localhost.

Przekazywanie poświadczeń między serwerami - LAB

1. Wyświetl polecenia pozwalające na modyfikację i wyświetlania opcji związanych z CREDSSP

2. Zezwól na przekazywanie poświadczeń:

a. Uruchom gpedit.msc

b. Przejdź do Local Security Policy >> Computer Configuration >> Administrative Templates >> System >> Credentials delegation >> Allow delegating fresh credentials with NTLM-only server authorization

c. Przełącz stan na Enabled

d. Kliknij “Show” obok Add servers to list I wpisz wsman/<nazwa twojego komputera>

3. Sprawdź czy delegowanie poświadczeń jest wyłączone. Gdyby delegowanie było włączone, wyłącz je.

4. W tym zadaniu symulujemy sytuację połączenia z serwera A do B i dalej do C. Aby zadanie można było wykonać nawet dysponując jednym komputerem połączenie jest nawiązywane z lokalnego komputera do lokalnego komputera a następnie znowu do tego samego lokalnego komputera. Otwórz połączenie do swojego komputera z wykorzystaniem Enter-PSSession

5. Mając otwarte połączenie zdalne, w ramach tej zdalnej sesji uruchom kolejne polecenie remotingu (do tego samego komputera): Invoke-Command wyświetlające datę i czas. Polecenie powinno zakończyć się błędem, bo nie jest jeszcze skonfigurowane przekazywanie poświadczeń.

6. Zamknij otwarte połączenie zdalne.

7. Skonfiguruj komputer do przekazywania poświadczeń:

a. Jako klient do przekazywania poświadczeń do komputera zdalnego

b. Jako serwer do przyjmowania poświadczeń

8. Sprawdź bieżącą konfigurację CredSSP

9. Połącz się do komputera z wykorzystaniem Enter-PSSession przekazując parametr -Authentication CredSSP oraz -Credential (Get-Credential)

10. W sesji zdalnej wykonaj ponownie polecenie Invoke-Command jak w pkt. (5). Tym razem polecenie powinno zadziałać. Potem zakończ sesję zdalną

Sesje - LAB

1. W tym zadaniu chodzi o skorelowanie pewnych czynności które mają być wykonywane w określonej kolejności na kilku maszynach. My oczywiście ze względu na brak wielu komputerów wykonamy wszystko na localhost ale zastosujemy logikę, która może być przeniesiona na inne komputery:

a. Utwórz obiekt sesji $sessionComp1 wskazujący na localhost

b. Utwórz obiekt sesji $sessionComp2 wskazujący na localhost

2. Wyświetl dostępne w tej chwili sesje

3. Korzystając z sesji do pierwszego komputera zapytaj o stan usługi bits i zapisz go w zdalnej zmiennej $bitStatus

4. Do lokalnej zmiennej $bitStatusComp1 wpisz wartość ze zmiennej z poprzedniego punktu (skorzystaj z właściwości Value zwracanej przez Invoke-Command)

5. Korzystając z sesji do drugiego komputera zapytaj o stan usługi wsearch i zapisz go w zdalnej zmiennej $wsearchStatus

6. Do lokalnej zmiennej $wsearchStatusComp2 wpisz wartość ze zmiennej z poprzedniego punktu (skorzystaj z właściwości Value zwracanej przez Invoke-Command)

7. Jeśli jednocześnie są spełnione warunki:

a. Usługa bits była w statusie ‘stopped’

b. Usługa wsearch była w statusie ‘running’

Wykonaj następujące czynności:

c. Uruchom usługę bits w sesji numer 1

d. Zatrzymaj usługę wsearch w sesji numer 2

8. Usuń wszystkie sesje.

Cz. 11

Wartości domyślne parametrów i plik
profile - LAB

1. Załóżmy, że planujesz wielokrotnie wykonywać polecenie Get-Service dla usługi wsearch. Dodaj odpowiedni parametr domyślny:

a. Polecenie: Get-Service

b. Parametr: Name

c. Wartość: wsearch

2. Przetestuj działanie polecenia Get-Service bez dodatkowych parametrów

3. Utwórz (z poziomu powershell) katalog: c:\users\<nazwa użytkownika>Documents\WindowsPowerShell

4. Utwórz (z poziomu powershell) plik profile.ps1 we wcześniej wymienionym katalogu

5. Wyedytuj (np. w notatniku) ten plik

6. Umieść w nim (poniższe polecenia należy oczywiście testować najpierw w powershell:

a. Powitanie użytkownika – skorzystaj z $env:username

b. Informację o nazwie komputera na jakim pracujesz – skorzystaj ze zmiennej $env:computername

c. Utwórz wirtualny napęd TEMP: zmapowany do katalogu c:\temp

7. Upewnij się, że execution policy pozwala na uruchamianie skryptów, a jeśli nie to zmień execution policy na RemoteSigned

8. Otwórz nową sesję powershell (np. w ISE File >> New PowershellTab).

Upewnij się, że profil został uruchomiony poprawnie.

Planowanie zadań - LAB

1. Chcesz w tle wykonać (długotrwałe) zadanie. Przygotuj polecenie, które z wykorzystaniem Start-Job:

a. Pobierze obiekt win32_LogicalDisk

b. Zapisze wynik w pliku c:\temp\disks.txt

2. Przygotuj polecenie, które w tle uruchomi polecenie pobierające i=obiekt WMI odpowiedzialny za procesor. Skorzystaj przy tym wyłącznie z polecenia Get-WMIObject i jego opcji.

3. Przygotuj polecenie, które na komputerze zdalnym uruchomi Get- Service. Skorzystaj przy tym z polecenie Invoke-Command i jego opcji.

4. Wyświetl listę jobów uruchamianych w tle

5. Pobierz wynik job-a zdefiniowanego w kroku (2) tak aby wynik można było odczytywać jeszcze kolejny raz

6. Pobierz wynik wszystkich jobów. Zrób to w jednym poleceniu

7. Usuń jednym poleceniem wszystkie joby

8. Przygotuj trigger do uruchomienia zadania zaplanowanego o godzinie

14:00 codziennie (możesz dopasować godzinę do np. za 10 minut od teraz)

9. Zarejestruj planowane zadanie, które zostanie wyzwolone triggerem z poprzedniego punktu, które pobierze informacje o dyskach logicznych i zapisze je w pliku w c:\ temp

10. Wyświetl listę zdefiniowanych zadań

11. Poczekaj aż zadanie się wykonana a potem wyrejestruj je