Zabezpieczanie dostęp do danych przy pomocy hasła

Przygotowanie:

```
# Pobierz ścieżkę do pulpitu bieżącego użytkownika
$desktopPath = [System.IO.Path]::Combine([System.Environment]::GetFolderPath("Desktop"))
# Ścieżka do nowego folderu "archiwum" na pulpicie
$archiveFolderPath = [System.IO.Path]::Combine($desktopPath, "archiwum")
# Upewnij się, że katalog istnieje, jeśli nie, utwórz go
if (-Not (Test-Path -Path $archiveFolderPath)) {
    New-Item -ItemType Directory -Path $archiveFolderPath
    Write-Output "Katalog 'archiwum' został utworzony na pulpicie."
} else {
    Write-Output "Katalog 'archiwum' już istnieje na pulpicie."
}
# Ścieżka do katalogu testowego, w którym chcesz utworzyć pliki
$folderPath = "C:\Test"
# Upewnij się, że katalog istnieje, jeśli nie, utwórz go
if (-Not (Test-Path -Path $folderPath)) {
    New-Item -ItemType Directory -Path $folderPath
}
cd C:\Test
fsutil file createnew smallfile.txt 1024
fsutil file createnew mediumfile.txt 1048576
fsutil file createnew largefile.txt 10485760
# Ścieżka do katalogu testowego, w którym chcesz utworzyć pliki
$folderPath = "C:\InnyTest"
# Upewnij się, że katalog istnieje, jeśli nie, utwórz go
if (-Not (Test-Path -Path $folderPath)) {
    New-Item -ItemType Directory -Path $folderPath
}
cd C:\InnyTest
```

```
fsutil file createnew maly.txt 1024
fsutil file createnew sredni.txt 1048576
fsutil file createnew duzy.txt 10485760
# Tworzenie katalogu C:\Docel
$folderPath = "C:\Docel"
# Upewnij się, że katalog istnieje, jeśli nie, utwórz go
if (-Not (Test-Path -Path $folderPath)) {
    New-Item -ItemType Directory -Path $folderPath
    Write-Output "Katalog $folderPath został utworzony."
} else {
    Write-Output "Katalog $folderPath już istnieje."
}
```

1. Jak stworzyć za pomocą PowerShell archiwum zip?

PowerShell v5.0 dodaje polecenia cmdlet Compress-Archive i Expand-Archive.

a) Wykonaj kompresje i dekompresję wybranego archiwum.

```
# Utwórz plik zip z zawartością C:\Test\
Compress-Archive -Path C:\Test -DestinationPath C:\Users\admin\Desktop\archiwum\archive.zip
# Dodaj więcej plików do pliku zip
# (Istniejące pliki w pliku zip o tej samej nazwie są zastępowane)
Compress-Archive -Path C:\InnyTest\*.txt -Update -DestinationPath
C:\Users\admin\Desktop\archiwum\archive.zip
# Wypakuj plik zip do C:\Docel\
Expand-Archive -Path C:\Users\admin\Desktop\archiwum\archive.zip -DestinationPath C:\Docel
```

b) Alternatywa w postaci funkcji:

```
function ZipFiles( $zipfilename, $sourcedir )
{
   Add-Type -Assembly System.IO.Compression.FileSystem
   $compressionLevel = [System.IO.Compression.CompressionLevel]::Optimal
```

```
    [System.IO.Compression.ZipFile]::CreateFromDirectory($sourcedir, $zipfilename, $compressionLevel, $false)
}
```

Użyj funkcji

Aby użyć tej funkcji do utworzenia pliku ZIP z katalogu C:\Test i zapisać wynikowy plik ZIP jako C:\archive.zip, możesz postępować w następujący sposób:

Otwórz PowerShell.

Wklej definicję funkcji ZipFiles.

Wykonaj funkcję z odpowiednimi argumentami:

```
$zipfilename = "C:\archive.zip"
$sourcedir = "C:\Test"
ZipFiles -zipfilename $zipfilename -sourcedir $sourcedir
```

2. Utwórz za pomocą PowerShell archiwum zip chronione hasłem.

Pobierz program na swój system Windows - możesz go pobrać **stąd**.

Istnieje kilka różnych sposobów tworzenia plików zip w powershell, ale niewiele z nich pozwala na utworzenie takiego, który jest chroniony hasłem.

Opis przykładu rozwiązania. Funkcja, która używa 7zip do wykonania zip, ponieważ 7zip obsługuje używanie hasła do zipowania plików. Ten skrypt szuka pliku wykonywalnego 7zip (7z.exe) w domyślnych lokalizacjach instalacji, a jeśli nie zostanie znaleziony, użyje samodzielnego pliku wykonywalnego 7zip (7za.exe), jeśli znajduje się on w tym samym katalogu co skrypt powershell.

a) Zapisz (opis w pliku Opis szczegółowy przykładu rozwiązania Skrypt PowerShell.pdf), wyjaśnij i wykonaj poniższą przykładową funkcję.

```
function Write-ZipUsing7Zip([string]$FilesToZip, [string]$ZipOutputFilePath, [string]$Password, [ValidateSet('7z','zip','gzip','bzip2','tar','iso','udf')][string]$CompressionType = 'zip', [switch]$HideWindow)
{
    # Look for the 7zip executable.
    $pathTo32Bit7Zip = "C:\Program Files (x86)\7-Zip\7z.exe"
    $pathTo64Bit7Zip = "C:\Program Files\7-Zip\7z.exe"
    $THIS_SCRIPTS_DIRECTORY = Split-Path $script:MyInvocation.MyCommand.Path
    $pathToStandAloneExe = Join-Path $THIS_SCRIPTS_DIRECTORY "7za.exe"
    if (Test-Path $pathTo64Bit7Zip) { $pathTo7ZipExe = $pathTo64Bit7Zip }
    elseif (Test-Path $pathTo32Bit7Zip) { $pathTo7ZipExe = $pathTo32Bit7Zip }
```

```
    elseif (Test-Path $pathToStandAloneExe) { $pathTo7ZipExe = $pathToStandAloneExe }
    else { throw "Could not find the 7-zip executable." }

    # Delete the destination zip file if it already exists (i.e. overwrite it).
    if (Test-Path $ZipOutputFilePath) { Remove-Item $ZipOutputFilePath -Force }

    $windowStyle = "Normal"
    if ($HideWindow) { $windowStyle = "Hidden" }

    # Create the arguments to use to zip up the files.
    # Command-line argument syntax can be found at: http://www.dotnetperls.com/7-zip-examples
    $arguments = "a -t$CompressionType ""$ZipOutputFilePath"" ""$FilesToZip"" -mx9"
    if (!([string]::IsNullOrEmpty($Password))) { $arguments += " -p$Password" }

    # Zip up the files.
    $p = Start-Process $pathTo7ZipExe -ArgumentList $arguments -Wait -PassThru -WindowStyle
$windowStyle

    # If the files were not zipped successfully.
    if (!(($p.HasExited -eq $true) -and ($p.ExitCode -eq 0)))
    {
        throw "There was a problem creating the zip file '$ZipFilePath'."
    }
}
```

b) Wywołaj na dwa sposoby przygotowaną funkcję.

Poniżej przykłady jak wywołać przygotowaną funkcję:

```
Write-ZipUsing7Zip -FilesToZip "C:\InnyTest" -ZipOutputFilePath "C:\InnyTest.zip" -Password
"password123"

Write-ZipUsing7Zip "C:\Docel\*.txt" "C:\DocelTxtFiles.zip" -HideWindow
```

c) Wyjaśnij w zeszycie sposób wywołania przygotowanej funkcji.

Funkcja Write-ZipUsing7Zip jest używana do tworzenia archiwów ZIP z wykorzystaniem programu 7-Zip. Przyjrzyjmy się, jak wywołać tę funkcję na podstawie podanych przykładów:

Przykład 1:

```
Write-ZipUsing7Zip -FilesToZip "C:\InnyTest" -ZipOutputFilePath "C:\InnyTest.zip" -Password "password123"
```

Wyjaśnienie:

1. **-FilesToZip "C:\InnyTest"**:
   - -FilesToZip określa ścieżkę do pliku lub folderu, który ma zostać skompresowany.
   - W tym przypadku cały folder C:\InnyTest będzie skompresowany.
2. **-ZipOutputFilePath "C:\InnyTest.zip"**:
   - -ZipOutputFilePath określa ścieżkę, gdzie ma zostać zapisany plik ZIP.
   - Wynikowy plik ZIP zostanie zapisany jako C:\InnyTest.zip.
3. **-Password "password123"**:
   - -Password opcjonalnie ustawia hasło na archiwum ZIP.
   - Tutaj hasło do archiwum będzie password123.

Przykład 2:

```
Write-ZipUsing7Zip "C:\Docel\*.txt" "C:\DocelTxtFiles.zip" -HideWindow
```

Wyjaśnienie:

1. **"C:\Docel*.txt"**:
   - Pierwszy argument bez flagi wskazuje na pliki do skompresowania.
   - Tutaj wszystkie pliki tekstowe (*.txt) w folderze C:\Docel będą skompresowane.
2. **"C:\DocelTxtFiles.zip"**:
   - Drugi argument bez flagi określa ścieżkę, gdzie ma zostać zapisany plik ZIP.
   - Wynikowy plik ZIP zostanie zapisany jako C:\DocelTxtFiles.zip.
3. **-HideWindow**:
   - -HideWindow to flaga, która powoduje, że okno 7-Zip nie będzie widoczne podczas kompresji.
   - Przydatne, jeśli chcemy, aby operacja kompresji odbyła się w tle, bez interakcji z użytkownikiem.

Podsumowanie:

Funkcja Write-ZipUsing7Zip pozwala na elastyczne tworzenie archiwów ZIP, przyjmując różne argumenty i flagi:

- -FilesToZip: Określa pliki/foldery do skompresowania.
- -ZipOutputFilePath: Określa lokalizację zapisu pliku ZIP.
- -Password: Opcjonalnie ustawia hasło na archiwum.

- -HideWindow: Opcjonalnie ukrywa okno 7-Zip podczas kompresji.

W obu przykładach funkcja jest wywoływana z różnymi zestawami argumentów, aby pokazać jej wszechstronność.