Praca z hasłami, bezpiecznymi ciągami i poświadczeniami w Windows PowerShell

Wstęp

Zapisz:

Hasła w PowerShell mogą być przechowywane w różnych formach:

String - zwykłe ciągi tekstowe. Używane do przechowywania dowolnego tekstu mogą przechowywać hasła. Ciągi są niezabezpieczone, są przechowywane w pamięci jako zwykły tekst, a większość poleceń cmdlet nie akceptuje haseł w tej formie.

System.Security.SecureString - ten typ jest podobny do zwykłego ciągu, ale jego zawartość jest zaszyfrowana w pamięci. Wykorzystuje szyfrowanie odwracalne, dzięki czemu hasło może zostać odszyfrowane w razie potrzeby, ale tylko przez zleceniodawcę, który je zaszyfrował.

System.Management.Automation.PSCredential - PSCredential to klasa składająca się z nazwy użytkownika (ciągu) i hasła(Bezpieczny ciąg). Jest to typ wymagany przez większość poleceń cmdlet do określania poświadczeń.

Wykonaj:

Konwersja z jednego typu na inny, są następujące metody;

1. ***Utwórz bezpieczny ciąg***

a) Wpisz hasło w interaktywnym wierszu

```
$SecurePassword = Read-Host -Prompt "Enter password" -AsSecureString
```

Powyższe polecenie w PowerShellu służy do bezpiecznego wprowadzenia hasła przez użytkownika i przechowania go jako obiekt SecureString. Oto szczegółowe wyjaśnienie tego polecenia:

Read-Host: Jest to cmdlet w PowerShellu, który służy do odczytywania wejścia od użytkownika. Wyświetla podany tekst jako monit i czeka na wprowadzenie danych przez użytkownika.

-Prompt "Enter password": Jest to parametr cmdletu Read-Host, który określa tekst monitu, który będzie wyświetlany użytkownikowi. W tym przypadku monit to "Enter password".

-AsSecureString: Jest to parametr cmdletu Read-Host, który określa, że wprowadzone dane mają być traktowane jako SecureString. Oznacza to, że:

- Wprowadzone dane są maskowane (nie są widoczne na ekranie).
- Dane są przechowywane w zaszyfrowanej formie w pamięci, co zwiększa bezpieczeństwo.

$SecurePassword: Jest to zmienna, do której przypisany jest wynik operacji Read-Host. W tym przypadku wynik to obiekt SecureString, który zawiera wprowadzone hasło.

Przykład użycia i działania:

```
# Użytkownik zostanie poproszony o wprowadzenie hasła
$SecurePassword = Read-Host -Prompt "Enter password" -AsSecureString
# Wyświetlenie typu przechowywanego hasła (powinno to być SecureString)
$SecurePassword.GetType().Name
```

Podsumowując, powyższe polecenie pozwala na bezpieczne wprowadzenie hasła przez użytkownika, które następnie jest przechowywane w formie zaszyfrowanej jako obiekt SecureString. Dzięki temu hasło nie jest widoczne na ekranie ani przechowywane w pamięci w postaci zwykłego tekstu, co zwiększa bezpieczeństwo.

b) Konwertuj *z istniejącej zmiennej w postaci zwykłego tekstu*

Poniższe polecenia w PowerShellu służą do zamiany hasła w postaci zwykłego tekstu na obiekt SecureString. Wyjaśnienie każdego z tych poleceń:

Pierwsze polecenie:

Przypisanie hasła w postaci zwykłego tekstu do zmiennej $PlainPassword:

```
$PlainPassword = "P@ssw0rd"
```

W tej linii tworzymy zmienną $PlainPassword i przypisujemy jej wartość "P@ssw0rd", która jest hasłem w postaci zwykłego tekstu.

Drugie polecenie:

Konwersja hasła na SecureString:

```
$SecurePassword = $PlainPassword | ConvertTo-SecureString -AsPlainText -Force
```

$PlainPassword | ConvertTo-SecureString: To polecenie używa potoku (|) do przekazania zawartości zmiennej $PlainPassword jako wejście do cmdletu ConvertTo-SecureString.

ConvertTo-SecureString: Jest to cmdlet, który konwertuje dane wejściowe na obiekt SecureString. Domyślnie, SecureString służy do bezpiecznego przechowywania danych tekstowych, takich jak hasła.

-AsPlainText: Określa, że wejściowy tekst ($PlainPassword) jest w postaci zwykłego tekstu (plaintext). Ten parametr jest wymagany, gdy konwertujemy zwykły tekst na SecureString.

-Force: Ten parametr jest używany, aby wymusić konwersję nawet wtedy, gdy PowerShell normalnie wyświetla ostrzeżenie, że konwersja zwykłego tekstu na SecureString może być niebezpieczna, ponieważ zwykły tekst jest łatwo dostępny w pamięci.

Podsumowując, powyższe polecenia pobierają hasło w postaci zwykłego tekstu, a następnie konwertują je na obiekt SecureString, który przechowuje hasło w zaszyfrowanej formie w pamięci. Jest to przydatne do zwiększenia bezpieczeństwa przechowywania i operacji na hasłach w skryptach PowerShell.

2. Utwórz poświadczenia PS

Zakładając, że masz hasło w postaci SecureString w zmiennej $SecurePassword:

Pierwsze polecenie:

Przypisanie nazwy użytkownika do zmiennej $UserName:

```
$UserName = "Domain\User"
```

W tej linii tworzymy zmienną $UserName i przypisujemy jej wartość "Domain\User", która jest pełną nazwą użytkownika, zawierającą domenę i nazwę użytkownika.

Drugie polecenie:

worzenie obiektu PSCredential:

```
$Credentials = New-Object System.Management.Automation.PSCredential -ArgumentList $UserName, $SecurePassword
```

New-Object: Jest to cmdlet w PowerShellu, który tworzy nowe instancje obiektów .NET.

System.Management.Automation.PSCredential: Jest to typ obiektu, który reprezentuje bezpieczne poświadczenia, składające się z nazwy użytkownika i hasła (w postaci SecureString).

-ArgumentList: Ten parametr przekazuje listę argumentów do konstruktora obiektu PSCredential. W tym przypadku przekazujemy nazwę użytkownika ($UserName) i hasło ($SecurePassword).

Podsumowując, powyższe polecenia tworzą obiekt PSCredential, który bezpiecznie przechowuje nazwę użytkownika i hasło. Ten obiekt jest często używany w PowerShellu do uwierzytelniania w różnych kontekstach, takich jak łączenie się z zdalnymi serwerami, dostęp do zasobów sieciowych czy wykonywanie operacji, które wymagają uwierzytelnienia.

3. Wyodrębnij hasło z PSCredentials

Hasło można łatwo uzyskać z obiektu PSCredential za pomocą metody GetNetworkCredential:

```
$PlainPassword = $Credentials.GetNetworkCredential().Password
```

Powyższe polecenie w PowerShellu służy do uzyskania hasła w postaci zwykłego tekstu z obiektu PSCredential. Wyjaśnienie:

$Credentials: Jest to zmienna, która przechowuje obiekt typu PSCredential. Obiekty tego typu służą do bezpiecznego przechowywania pary nazwa użytkownika - hasło. Obiekt PSCredential zazwyczaj tworzony jest za pomocą cmdletu Get-Credential.

GetNetworkCredential(): Jest to metoda obiektu PSCredential, która zwraca obiekt typu NetworkCredential. Ten obiekt zawiera właściwości UserName, Password i Domain.

.Password: Jest to właściwość obiektu NetworkCredential, która zawiera hasło w postaci zwykłego tekstu.

Działa w następujący sposób:

- GetNetworkCredential() zwraca obiekt NetworkCredential z PSCredential.
- .Password uzyskuje hasło w formie zwykłego tekstu z obiektu NetworkCredential.
- Wynik jest przypisany do zmiennej $PlainPassword.

Przykład użycia:

```
# Tworzenie obiektu PSCredential (zostaniesz poproszony o wprowadzenie nazwy użytkownika i hasła)

$Credentials = Get-Credential

# Pobranie hasła w postaci zwykłego tekstu
```

```
$PlainPassword = $Credentials.GetNetworkCredential().Password

# Wyświetlenie hasła

$PlainPassword
```

4. Wyodrębnij hasło z SecureString

Jeśli masz tylko prosty SecureString z hasłem, możesz skonstruować obiekt PSCredentials i wyodrębnić hasło przy użyciu poprzedniej metody. Inna metoda to:

Poniższe polecenia w PowerShellu służą do zamiany hasła przechowywanego w obiekcie SecureString na zwykły tekst. Jest to przydatne w sytuacjach, gdy potrzebujesz użyć hasła w formacie zwykłego tekstu (np. do uwierzytelnienia w jakiejś usłudze). Wyjaśnienie każdego z tych poleceń:

SecureString to specjalny typ danych w .NET (a także w PowerShellu), który pozwala na bezpieczne przechowywanie danych tekstowych (takich jak hasła) w pamięci. Dzięki temu zawartość SecureString jest zaszyfrowana w pamięci, co utrudnia nieautoryzowany dostęp.

Pierwsze polecenie:

```
$BSTR = [System.Runtime.InteropServices.Marshal]::SecureStringToBSTR($SecurePassword)
```

- [System.Runtime.InteropServices.Marshal] to klasa w .NET, która zawiera metody do operacji na wskaźnikach i pamięci.
- SecureStringToBSTR to metoda, która zamienia obiekt SecureString na niezarządzany wskaźnik (BSTR - Binary String). Ten wskaźnik wskazuje na obszar pamięci, gdzie hasło jest przechowywane w formacie zwykłego tekstu.

Drugie polecenie:

```
$PlainPassword = [System.Runtime.InteropServices.Marshal]::PtrToStringAuto($BSTR)
```

- PtrToStringAuto to metoda, która zamienia wskaźnik (w tym przypadku BSTR) na zarządzany łańcuch znaków (String) w .NET.
- W wyniku tego polecenia, zmienna $PlainPassword zawiera hasło w formacie zwykłego tekstu.

Podsumowując, powyższe polecenia zamieniają bezpiecznie przechowywane hasło (jako SecureString) na zwykły tekst, co może być konieczne w niektórych scenariuszach. Ważne jest jednak, aby po użyciu

tego tekstowego hasła, odpowiednio je wyczyścić z pamięci, aby zminimalizować ryzyko nieautoryzowanego dostępu do tego hasła.

5. Zapisywanie zaszyfrowanego hasła do pliku lub rejestru

Jeśli potrzebujesz przechowywać hasło do skryptu, który działa w trybie nienadzorowanym przez harmonogram lub w inny sposób, możesz zapisać je w systemie plików lub rejestrze w postaci zaszyfrowanej. Przypomina to ciąg znaków reprezentujący SecureString. Tylko użytkownik, który utworzył tę linię, może ją odszyfrować i używać, więc podczas zapisywania tej wartości użyj tego samego konta, z którego będzie korzystać skrypt lub usługa.

Aby zapisać zaszyfrowane hasło do pliku lub rejestru w PowerShellu, można użyć ConvertFrom-SecureString do konwersji SecureString na zaszyfrowany ciąg znaków, a następnie zapisać ten ciąg do pliku. Oto przykład, jak to zrobić:

a) Zapisywanie zaszyfrowanego hasła do pliku

1. Utwórz SecureString z hasła:

```
$SecurePassword = Read-Host -Prompt "Enter password" -AsSecureString
```

2. Konwertuj SecureString na zaszyfrowany ciąg znaków:

```
$EncryptedPassword = $SecurePassword | ConvertFrom-SecureString
```

3. Zapisz zaszyfrowany ciąg znaków do pliku:

```
# Upewnij się, że ścieżka do folderu istnieje
$FolderPath = "C:\path\to"
if (-not (Test-Path -Path $FolderPath)) {
    New-Item -ItemType Directory -Path $FolderPath
}
# Przypisz pełną ścieżkę pliku
$FilePath = "C:\path\to\password.txt"
# Zapisz zaszyfrowany ciąg znaków do pliku
```

$EncryptedPassword | Out-File $FilePath

b) Odczytywanie zaszyfrowanego hasła z pliku i używanie go

1. Odczytaj zaszyfrowany ciąg znaków z pliku:

$EncryptedPassword = Get-Content $FilePath

2. Konwertuj zaszyfrowany ciąg znaków z powrotem na SecureString:

$SecurePassword = $EncryptedPassword | ConvertTo-SecureString

3. Utwórz obiekt PSCredential:

$UserName = "Domain\User"

$Credentials = New-Object System.Management.Automation.PSCredential -ArgumentList $UserName, $SecurePassword

Przykład pełnego skryptu:

Zapisywanie hasła do pliku

Krok 1: Utwórz SecureString z hasła

$SecurePassword = Read-Host -Prompt "Enter password" -AsSecureString

Krok 2: Konwertuj SecureString na zaszyfrowany ciąg znaków

$EncryptedPassword = $SecurePassword | ConvertFrom-SecureString

Krok 3: Zapisz zaszyfrowany ciąg znaków do pliku

$FilePath = "C:\path\to\password.txt"

$EncryptedPassword | Out-File $FilePath

Odczytywanie hasła z pliku i używanie go

Krok 1: Odczytaj zaszyfrowany ciąg znaków z pliku

$FilePath = "C:\path\to\password.txt"

$EncryptedPassword = Get-Content $FilePath

Krok 2: Konwertuj zaszyfrowany ciąg znaków z powrotem na SecureString

$SecurePassword = $EncryptedPassword | ConvertTo-SecureString

```
# Krok 3: Utwórz obiekt PSCredential

$UserName = "Domain\User"

$Credentials = New-Object System.Management.Automation.PSCredential -ArgumentList $UserName, $SecurePassword
```

Uwaga

- Upewnij się, że tylko użytkownik, który utworzył zaszyfrowane hasło, ma dostęp do pliku zawierającego zaszyfrowany ciąg znaków.
- Plik powinien być przechowywany w bezpiecznym miejscu z odpowiednimi uprawnieniami, aby zapobiec nieautoryzowanemu dostępowi.

c) Zapisywanie zaszyfrowanego hasła w rejestrze

Możesz także użyć rejestru do przechowywania zaszyfrowanego hasła. Przykłady:

Zapisywanie hasła do rejestru

```
# Utwórz SecureString z hasła

$SecurePassword = Read-Host -Prompt "Enter password" -AsSecureString

# Konwertuj SecureString na zaszyfrowany ciąg znaków

$EncryptedPassword = $SecurePassword | ConvertFrom-SecureString

# Zapisz zaszyfrowany ciąg znaków do rejestru

$RegistryPath = "HKCU:\Software\MyScript"

New-Item -Path $RegistryPath -Force | Out-Null

Set-ItemProperty -Path $RegistryPath -Name "EncryptedPassword" -Value $EncryptedPassword
```

Odczytywanie hasła z rejestru i używanie go

```
# Odczytaj zaszyfrowany ciąg znaków z rejestru

$RegistryPath = "HKCU:\Software\MyScript"

$EncryptedPassword = Get-ItemProperty -Path $RegistryPath -Name "EncryptedPassword"

$EncryptedPassword = $EncryptedPassword.EncryptedPassword
```

```
# Konwertuj zaszyfrowany ciąg znaków z powrotem na SecureString
$SecurePassword = $EncryptedPassword | ConvertTo-SecureString
# Utwórz obiekt PSCredential
$UserName = "Domain\User"
$Credentials = New-Object System.Management.Automation.PSCredential -ArgumentList $UserName, $SecurePassword
```

Te metody pozwalają na bezpieczne przechowywanie i odczytywanie hasła, które może być używane w skryptach działających w trybie nienadzorowanym.

6. Konwertowanie zmiennej SecureString na bezpieczną reprezentację w postaci zwykłego tekstu

Poniższe wyjaśnienie dotyczy procesu konwersji obiektu SecureString na zaszyfrowany ciąg znaków, który można przechowywać w pliku, rejestrze lub innym miejscu, a następnie jego odszyfrowania na tym samym komputerze i koncie użytkownika. Wyjaśnienie:

a) Konwersja SecureString na zaszyfrowany ciąg znaków

```
$SecureStringAsPlainText = $SecurePassword | ConvertFrom-SecureString
```

$SecurePassword: Jest to obiekt SecureString, który zawiera hasło w zaszyfrowanej formie.

ConvertFrom-SecureString: Cmdlet konwertuje SecureString na zaszyfrowany ciąg znaków w formacie, który można łatwo przechowywać jako tekst. Wynikowy ciąg znaków wygląda jak ciąg heksadecymalny, np. „ea32f9d30de3d3dc7fcd86a6a8f587ed9” (w rzeczywistości jest znacznie dłuższy).

b) Przechowywanie zaszyfrowanego ciągu znaków

Ten zaszyfrowany ciąg znaków można przechowywać w pliku, rejestrze lub innym miejscu:

```
$SecureStringAsPlainText | Out-File -FilePath "C:\path\to\password.txt"
```

c) Odszyfrowanie ciągu znaków

Odszyfrowanie zaszyfrowanego ciągu znaków z powrotem na SecureString:

```
$SecureStringAsPlainText = Get-Content -Path "C:\path\to\password.txt"
$SecureString = $SecureStringAsPlainText | ConvertTo-SecureString
```

Get-Content: Cmdlet odczytuje zawartość pliku, w którym przechowywany jest zaszyfrowany ciąg znaków.

ConvertTo-SecureString: Cmdlet konwertuje zaszyfrowany ciąg znaków z powrotem na obiekt SecureString.

d) Ograniczenia i zabezpieczenia

Ograniczenia: Odszyfrowanie działa tylko na tym samym komputerze i pod tym samym kontem użytkownika, które zaszyfrowało ciąg znaków. Jeśli inny użytkownik na tym samym komputerze lub ten sam użytkownik na innym komputerze spróbuje odszyfrować ciąg, otrzyma błąd:

convertto-securestring : Key not valid for use in specified state.

Bezpieczeństwo: Ten mechanizm zapewnia, że tylko autoryzowany użytkownik na określonym komputerze może odszyfrować przechowywane hasło, co zwiększa bezpieczeństwo przechowywania danych uwierzytelniających.

Przykład pełnego skryptu

Zapisywanie zaszyfrowanego hasła do pliku:

```
# Krok 1: Utwórz SecureString z hasła

$SecurePassword = Read-Host -Prompt "Enter password" -AsSecureString

# Krok 2: Konwertuj SecureString na zaszyfrowany ciąg znaków

$SecureStringAsPlainText = $SecurePassword | ConvertFrom-SecureString

# Krok 3: Zapisz zaszyfrowany ciąg znaków do pliku

$FilePath = "C:\path\to\password.txt"

$SecureStringAsPlainText | Out-File $FilePath
```

Odczytywanie hasła z pliku i używanie go:

```
# Krok 1: Odczytaj zaszyfrowany ciąg znaków z pliku

$FilePath = "C:\path\to\password.txt"

$SecureStringAsPlainText = Get-Content $FilePath

# Krok 2: Konwertuj zaszyfrowany ciąg znaków z powrotem na SecureString
```

```
$SecureString = $SecureStringAsPlainText | ConvertTo-SecureString

# Krok 3: Utwórz obiekt PSCredential

$UserName = "Domain\User"

$Credentials = New-Object System.Management.Automation.PSCredential -ArgumentList $UserName, $SecureString
```

Powyższe skrypty pokazują, jak bezpiecznie przechowywać i odczytywać hasła w skryptach PowerShell, działających w trybie nienadzorowanym.

Zapoznaj się z poniższymi uwagami i wykonaj notatkę (zapisz).

Najlepsze praktyki

1. Jeśli to możliwe, nie pytaj o hasła i spróbuj użyć zintegrowanego uwierzytelniania systemu Windows.
2. Gdy nie jest to możliwe lub gdy przydatne jest określenie innych poświadczeń, polecenia cmdlet powinny akceptować hasła tylko w postaci PSCredentials lub (jeśli nazwa użytkownika nie jest potrzebna) jako SecureString, ale nie w postaci zwykłego tekstu.
3. Jeśli musisz poprosić użytkownika o poświadczenia, użyj polecenia cmdlet Get-Credential. Wykorzystuje standardową funkcję systemu Windows do odbierania hasła w spójny i bezpieczny sposób bez przechowywania go w pamięci jako zwykłego tekstu.
4. Poświadczenia powinny być przekazywane do systemu zewnętrznego również w możliwie najbezpieczniejszy sposób, najlepiej również jako PSCredentials.
5. Hasło nie powinno być zapisywane na dysku, w rejestrze lub innym niechronionym magazynie jako zwykły tekst. Jeśli to możliwe, użyj reprezentacji SecureString w postaci zwykłego tekstu.