Helion

packt

PowerShell w cyberbezpieczeństwie

Automatyzacja zadań, tworzenie skryptów, hakowanie i obrona. Red Team kontra Blue Team

MIRIAM C. WIESNER
PRZEDMOWA: TANYA JANCA

Tytuł oryginału: PowerShell Automation and Scripting for Cybersecurity: Hacking and defense for red and blue teamers

Tłumaczenie: Grzegorz Kowalczyk

ISBN: 978-83-289-2225-9

Drogi Czytelniku!
Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres
helion.pl/user/opinie/powcyb
Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Helion S.A.
ul. Kościuszki 1c, 44-100 Gliwice
tel. 32 230 98 63
e-mail: *helion@helion.pl*
WWW: *helion.pl* (księgarnia internetowa, katalog książek)

Printed in Poland.

Spis treści

CZĘŚĆ 2. Odkrywamy tajemnice — AD/AAD, tożsamości, dostęp do systemu i codzienne zadania związane z bezpieczeństwem

CZĘŚĆ 3. Zabezpieczanie PowerShella — skuteczne metody ochrony

Technologie PowerShell Remote Management i PSRemoting

Rozdział 3

PowerShell, jako narzędzie do automatyzacji zadań administracyjnych, oferuje technologię **PowerShell Remoting** (**PSRemoting**), która jest kluczowa w zarządzaniu wieloma komputerami jednocześnie. Dzięki PSRemoting wystarczy jedno polecenie, aby uruchomić proces na setkach maszyn.

Ale podobnie jak jest w przypadku pracy z pojedynczymi komputerami, technologia PowerShell Remoting jest tylko tak bezpieczna, jak Twoja konfiguracja: jeżeli nie zamkniesz drzwi swojego domu, włamywacze mogą się do niego dostać.

Tak samo jest w przypadku pracy z komputerami i PowerShell Remoting — jeżeli odpowiednio nie wzmocnisz konfiguracji i użyjesz słabych zabezpieczeń i ustawień, potencjalni napastnicy mogą to wykorzystać i użyć do przeprowadzenia skutecznego ataku.

W tym rozdziale poznasz nie tylko podstawy technologii PSRemoting oraz sposób włączania i konfigurowania tej usługi, ale także odkryjesz najlepsze praktyki dotyczące utrzymywania bezpiecznej konfiguracji PowerShell Remoting. Chociaż technologia PowerShell Remoting jest z natury dosyć bezpieczna, istnieje cały szereg dodatkowych mechanizmów i środków, które mogą jeszcze bardziej wzmocnić jej bezpieczeństwo. Dokładnie omówimy te ustawienia, aby pomóc w utrzymaniu bezpiecznej konfiguracji PSRemoting.

Zobaczymy również, jak wygląda ruch sieciowy PSRemoting w zależności od używanego protokołu uwierzytelniania. Na końcu dowiesz się, jak prawidłowo skonfigurować PowerShell Remoting, jakich ustawień unikać oraz jak wykorzystać tę technologię do efektywnego i bezpiecznego wykonywania poleceń na zdalnych komputerach.

W tym rozdziale poznasz następujące tematy:

- Praca zdalna z PowerShellem.
- Włączanie PowerShell Remoting.
- Konfiguracja punktów końcowych PowerShell (sesje).
- Uwierzytelnianie w PowerShell Remoting i kwestie bezpieczeństwa.
- Wykonywanie poleceń przy użyciu PowerShell Remoting.
- Praca z PowerShell Remoting.
- Najlepsze praktyki dotyczące PowerShell Remoting.

Wymagania techniczne

Aby w pełni wykorzystać zagadnienia opisywane w tym rozdziale, upewnij się, że posiadasz następujące elementy:

- PowerShell 7.3 lub nowszy.
- Visual Studio Code.
- Wireshark.
- Laboratorium testowe z kontrolerem domeny i co najmniej jedną maszyną testową.
- Dostęp do repozytorium GitHub dla rozdziału 3.: *https://github.com/PacktPublishing/PowerShell-Automation-and-Scripting-for-Cybersecurity/tree/master/Chapter03*.

Praca zdalna z PowerShellem

PowerShell został stworzony z myślą o automatyzacji zadań związanych z zarządzaniem komputerami oraz ułatwieniu pracy administratorom systemów. Zdalne zarządzanie było częścią tej koncepcji od samego początku, co zostało opisane przez Jeffreya Snovera w Monad Manifesto z 2002 roku: *https://www.jsnover.com/blog/2011/10/01/monad-manifesto/*. Aby przyspieszyć wydanie wersji 1.0, niektóre funkcje, w tym PSRemoting, zostały wprowadzone dopiero w późniejszych aktualizacjach. PSRemoting oficjalnie pojawił się w wersji 2.0, a następnie został udoskonalony w wersji 3.0.

PSRemoting szybko stał się jedną z kluczowych funkcji PowerShell, stanowiąc fundament dla wielu innych funkcji, takich jak przepływy pracy.

Choć PSRemoting współpracuje z różnymi metodami uwierzytelniania, domyślnym protokołem dla uwierzytelniania w domenach jest Kerberos, który jest najbezpieczniejszym i najczęściej wykorzystywanym protokołem w środowiskach Active Directory — możemy śmiało założyć, że najprawdopodobniej jest wykorzystywany przez większość użytkowników PSRemoting. Jeżeli Kerberos nie jest dostępny, PSRemoting przechodzi na NTLM, umożliwiając uwierzytelnianie w grupach roboczych.

Windows PowerShell wspiera zdalne zarządzanie dzięki różnym technologiom. Standardowo PSRemoting korzysta z **Windows Remote Management (WinRM)** jako protokołu transportowego. Warto jednak pamiętać, że WinRM to tylko jeden z kilku dostępnych protokołów do obsługi zdalnego zarządzania w PowerShellu. PSRemoting to specyficzny protokół **(PSRP)**, który reguluje sposób przetwarzania danych wejściowych, wyjściowych, strumieni danych, serializacji obiektów i innych. PSRP może działać z różnymi protokołami transportowymi, w tym **WS-Management (WS-Man)**, **Secure Shell (SSH)**, **Hyper-V VMBus** i innymi. Choć **Windows Management Instrumentation (WMI)** i **Remote Procedure Call (RPC)** to technologie zdalnego zarządzania wspierające narzędzie PowerShell, nie są one częścią protokołu PSRemoting.

Różnice pomiędzy tymi technologiami zdalnych połączeń i zdalnego zarządzania znajdują również odzwierciedlenie w używanych przez nie protokołach komunikacyjnych, tak jak pokazano w tabeli 3.1.

Tabela 3.1. Przegląd metod połączeń i używanych protokołów

Metoda połączenia zdalnego	Wykorzystywany protokół
PowerShell Remoting via WinRM (domyślnie)	WS-Management
WMI	DCOM/RPC
Cmdlety CIM	WS-Management
SSH Remoting	SSH

PSRemoting jest włączony tylko w systemach Windows Server 2012 R2 i nowszych wersjach, przy czym domyślnie dostęp do połączeń mają tylko członkowie grupy *Administratorzy*. Z kolei PowerShell Core oferuje wsparcie dla różnych protokołów zdalnego zarządzania, takich jak WMI, Web-Services Management (WS-Management) oraz SSH Remoting. Warto jednak zaznaczyć, że PowerShell Core nie obsługuje połączeń za pomocą RPC.

PowerShell Remoting przy użyciu WinRM

DMTF (dawniej znana jako *Distributed Management Task Force*) to organizacja non profit, która opracowuje otwarte standardy zarządzania, takie jak Common Information Model (CIM) oraz WS-Management.

WS-Management definiuje protokół oparty na protokole **SOAP** (*Simple Object Access Protocol*), który umożliwia zarządzanie serwerami i usługami internetowymi. Implementacją WS-Management w systemach firmy Microsoft jest **WinRM**.

Przy próbie nawiązania połączenia PSRemoting klient WinRM wysyła komunikaty SOAP za pomocą protokołu WS-Management, wykorzystując protokoły **HTTP** lub **HTTPS**.

Kiedy PSRemoting używa WinRM, nasłuchuje na następujących portach:

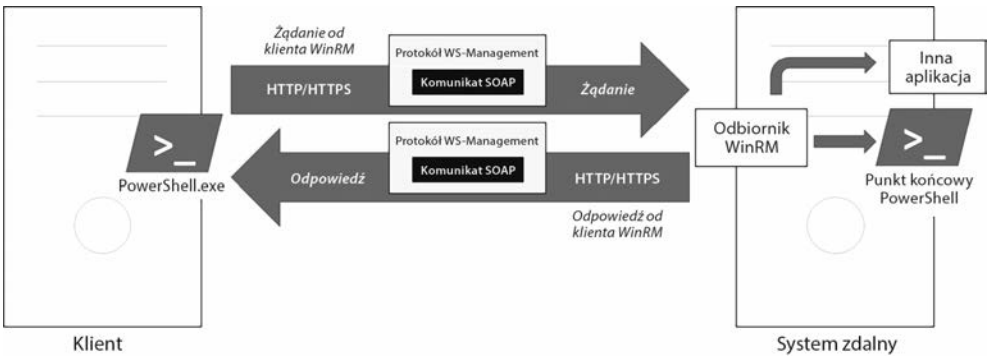
- **HTTP**: 5985,
- **HTTPS**: 5986.

Niezależnie od tego, czy używany jest protokół HTTP, czy HTTPS, cały ruch PSRemoting jest zawsze szyfrowany po zakończeniu procesu uwierzytelniania, zgodnie z wybranym protokołem uwierzytelniania. Więcej szczegółowych informacji na temat różnych protokołów uwierzytelniania znajdziesz w podrozdziale „Uwierzytelnianie".

Na zdalnym hoście uruchomiona jest usługa WinRM, która jest skonfigurowana do obsługi jednego lub więcej odbiorników — zarówno dla http, jak i HTTPS. Każdy z tych odbiorników oczekuje na przychodzący ruch HTTP/HTTPS przekazywany przez protokół WS-Management.

Po odebraniu ruchu usługa WinRM ustala, do którego punktu końcowego PowerShell lub aplikacji ruch jest przeznaczony i przekazuje go dalej, tak jak pokazano na rysunku 3.1.

Aby ułatwić zrozumienie działania WinRM, ten diagram został nieco uproszczony. W rzeczywistości proces `PowerShell.exe` nie jest wykorzystywany; zamiast niego uruchamiany jest proces `Wsmprovhost.exe`, który obsługuje połączenia PSRemoting.



Rysunek 3.1. Zastosowanie WinRM i WS-Management do połączeń za pośrednictwem PSRemoting

Ponieważ WinRM i WS-Management są domyślnymi protokołami przy nawiązywaniu połączeń zdalnych, w tym rozdziale skoncentrujemy się głównie na tych technologiach, ale dla pełnego obrazu omówię również pokrótce inne dostępne technologie zdalnego zarządzania.

Jeżeli chcesz dowiedzieć się czegoś więcej na temat protokołów WinRM i WS-Management, powinieneś zajrzeć do następujących źródeł:

- *https://docs.microsoft.com/en-us/windows/win32/winrm/windows-remote-management-architecture*,
- *https://github.com/devops-collective-inc/secrets-of-powershell-remoting*.

Windows Management Instrumentation (WMI) i Common Information Model (CIM)

WMI jest implementacją CIM firmy Microsoft, otwartego standardu zaprojektowanego przez DMTF.

WMI został wprowadzony wraz z Windows NT 4.0 i został włączony do systemu operacyjnego Windows, począwszy od Windows 2000. Nadal jest obecny we wszystkich nowoczesnych systemach, w tym Windows 10 i Windows Server 2019.

CIM określa sposób, w jaki elementy systemu IT są reprezentowane jako obiekty oraz jak poszczególne obiekty są ze sobą powiązane, co umożliwia efektywne zarządzanie systemami IT, niezależnie od producenta czy platformy.

WMI opiera się na technologii **Distributed Component Object Model (DCOM)** oraz protokole RPC (*Remote Procedure Call*), który jest kluczowym mechanizmem komunikacji w ramach modelu DCOM.

DCOM został zaprojektowany tak, aby umożliwić komunikację w modelu **Component Object Model (COM)** przez sieć i jest poprzednikiem technologii .NET Remoting.

W tej sekcji przedstawione zostaną jedynie podstawowe informacje na temat poleceń cmdlet związanych z WMI i CIM, aby ułatwić zrozumienie technologii zdalnego zarządzania

omówionych w tym rozdziale. Więcej szczegółowych informacji na temat technologii COM, WMI i CIM znajdziesz w rozdziale 5., „PowerShell to narzędzie o ogromnych możliwościach — dostęp do systemu i interfejsów API".

Cmdlety WMI

Począwszy od wersji PowerShell Core 6, polecenia cmdlet WMI zostały uznane za przestarzałe i nie powinny być używane w nowszych wersjach PowerShella. Warto jednak pamiętać, że są one nadal obsługiwane w starszych wersjach, takich jak PowerShell 5.1 w systemie Windows 10, i będą nadal obsługiwane przez cały okres wsparcia tych systemów operacyjnych. Jeżeli to jednak możliwe, zamiast z nich powinieneś korzystać z nowszych cmdletów CIM, które są kompatybilne zarówno z systemami operacyjnymi Windows, jak i innymi.

Zanim przejdziemy do pracy z przestarzałymi, ale wciąż dostępnymi poleceniami cmdlet WMI, warto dowiedzieć się, jak można je znaleźć.

Aby znaleźć wszystkie polecenia cmdlet i funkcje, które mają w nazwie ciąg znaków *wmi*, możesz użyć polecenia `Get-Command`. Za pomocą parametru `-CommandType` możesz określić, jakiego rodzaju poleceń szukasz. W przykładzie pokazanym poniżej będziemy szukać poleceń cmdlet i funkcji:

```
> Get-Command -Name *wmi* -CommandType Cmdlet,Function
CommandType     Name                Version     Source
-----------     ----                -------     ------
Cmdlet          Get-WmiObject       3.1.0.0     Microsoft.PowerShell.Management
Cmdlet          Invoke-WmiMethod    3.1.0.0     Microsoft.PowerShell.Management
Cmdlet          Register-WmiEvent   3.1.0.0     Microsoft.PowerShell.Management
Cmdlet          Remove-WmiObject    3.1.0.0     Microsoft.PowerShell.Management
Cmdlet          Set-WmiInstance     3.1.0.0     Microsoft.PowerShell.Management
```

Przykładem cmdlet WMI jest polecenie `Get-WmiObject`. Za pomocą tego polecenia można wysyłać zapytania do komputerów lokalnych i zdalnych.

Aby wyświetlić listę wszystkich klas WMI dostępnych na Twoim komputerze, możesz użyć parametru `-List`, tak jak pokazano w kolejnym przykładzie:

```
> Get-WmiObject -List
NameSpace: ROOT\cimv2
Name                        Methods     Properties
----                        -------     ----------
CIM_Indication              {}          {CorrelatedIndications, IndicationFilterName,
                                        IndicationIde...
CIM_ClassIndication         {}          {ClassDefinition, CorrelatedIndications,
                                        IndicationFilterNa...
CIM_ClassDeletion           {}          {ClassDefinition, CorrelatedIndications,
                                        IndicationFilterNa...
...
```

A oto przykład użycia polecenia `Get-WmiObject` do wyświetlenia informacji o usługach Windows działających na Twoim komputerze lokalnym:

```
> Get-WmiObject -Class Win32_Service
ExitCode     : 0
```

```
Name          : AdobeARMservice
ProcessId     : 3556
StartMode     : Auto
State         : Running
Status        : OK
...
```

Zapytanie takie można wysłać nie tylko do komputera lokalnego, ale także do komputera zdalnego. Aby to zrobić, powinieneś użyć parametru `-ComputerName`, po którym następuje nazwa komputera zdalnego. Kolejny przykład pokazuje, jak pobrać te same informacje z komputera zdalnego o nazwie `PSSec-PC02`:

```
> Get-WmiObject -Class Win32_Service -ComputerName PSSec-PC02
```

Powyższe polecenie zwraca listę wszystkich usług dostępnych na tym zdalnym komputerze.

Za pomocą parametru `-Query` można zdefiniować zapytanie, które zostanie uruchomione w bazie danych CIM podanego komputera. Polecenie pokazane poniżej wyświetla wszystkie usługi o nazwie WinRM:

```
> Get-WmiObject -ComputerName PSSec-PC02 -Query "select * from win32_service where
↪name='WinRM'"
ExitCode      : 0
Name          : WinRM
ProcessId     : 6408
StartMode     : Auto
State         : Running
Status        : OK
```

W tym przykładzie wykonujemy zapytanie `select * from win32_service where name='WinRM'` na komputerze zdalnym o nazwie `PSSec-PC02`.

Za pomocą cmdletów PowerShell WMI można również wywoływać metody WMI, usuwać obiekty i wykonywać wiele innych operacji.

Czy wiesz, że...

Protokół RPC, który stanowi podstawę WMI, przestał być wspierany w PowerShell Core 6. Wynika to z dążenia PowerShella do kompatybilności międzyplatformowej: począwszy od wersji PowerShell 7, protokół RPC jest obsługiwany wyłącznie na komputerach z systemem Windows.

Cmdlety CIM

Wraz z wydaniem PowerShell 3.0, które towarzyszyło Windows Server 2012 i Windows 8, wprowadzono nowy zestaw poleceń cmdlet do zarządzania obiektami, zgodny ze standardami CIM i WS-Man.

W przeszłości polecenia cmdlet WMI zaczęły odchodzić od standardów DMTF, co uniemożliwiało zarządzanie międzyplatformowe. Aby temu zaradzić, firma Microsoft przywróciła zgodność ze standardami DMTF CIM, wprowadzając nowe cmdlety CIM.

Aby znaleźć wszystkie cmdlety związane z CIM, możesz użyć polecenia `Get-Command`:

```
> Get-Command -Name "*cim*" -CommandType Cmdlet,Function
CommandType     Name                             Version     Source
-----------     ----                             -------     ------
Cmdlet          Get-CimAssociatedInstance        1.0.0.0     CimCmdlets
Cmdlet          Get-CimClass                     1.0.0.0     CimCmdlets
Cmdlet          Get-CimInstance                  1.0.0.0     CimCmdlets
Cmdlet          Get-CimSession                   1.0.0.0     CimCmdlets
Cmdlet          Invoke-CimMethod                 1.0.0.0     CimCmdlets
Cmdlet          New-CimInstance                  1.0.0.0     CimCmdlets
Cmdlet          New-CimSession                   1.0.0.0     CimCmdlets
Cmdlet          New-CimSessionOption             1.0.0.0     CimCmdlets
Cmdlet          Register-CimIndicationEvent      1.0.0.0     CimCmdlets
Cmdlet          Remove-CimInstance               1.0.0.0     CimCmdlets
Cmdlet          Remove-CimSession                1.0.0.0     CimCmdlets
Cmdlet          Set-CimInstance                  1.0.0.0     CimCmdlets
```

W powyższym przykładzie szukamy wszystkich poleceń cmdlet i funkcji, które mają w nazwie ciąg znaków *cim*.

Pełną listę dostępnych poleceń cmdlet CIM do interakcji z serwerami CIM można znaleźć na stronie *https://learn.microsoft.com/pl-pl/powershell/module/cimcmdlets/*.

Open Management Infrastructure (OMI)

W celu wsparcia zarządzania wieloplatformowego firma Microsoft opracowała w 2012 roku środowisko **Open Management Infrastructure (OMI)** (*https://github.com/Microsoft/omi*), które jednak nigdy nie zyskało dużej popularności i obecnie nie jest szeroko stosowane. Z tego powodu Microsoft postanowił wprowadzić obsługę zdalnego zarządzania przy użyciu SSH.

PowerShell Remoting przy użyciu SSH

Aby umożliwić PSRemoting pomiędzy hostami Windows i Linux, firma Microsoft wprowadziła obsługę PSRemoting przez **SSH** w PowerShell 6.

Wymagania PSRemoting via SSH

Aby używać PSRemoting przez SSH, należy zainstalować *PowerShell w wersji 6* lub wyższej oraz *SSH* na wszystkich komputerach. Od Windows 10 w wersji 1809 i Windows Server 2019, OpenSSH dla Windows jest już zintegrowany z systemem operacyjnym.

PowerShell Remoting w systemie Linux

Aby rozpocząć pracę z PowerShellem w systemie Linux, musisz najpierw zainstalować PowerShell Core, wykonując kroki opisane w jego oficjalnej dokumentacji, dostosowanej do konkretnej dystrybucji systemu Linux: *https://docs.microsoft.com/pl-pl/powershell/scripting/install/installing-powershell-core-on-linux*.

W moim laboratorium demonstracyjnym mam serwer z zainstalowanym systemem Debian 10. Pamiętaj, że niektóre kroki opisywane w przykładach mogą się nieco różnić w zależności od Twojego systemu operacyjnego.

Otwórz plik konfiguracyjny */etc/ssh/sshd_config* do edycji za pomocą wybranego edytora tekstu. W moim przykładzie używam edytora `vi`:

```
> vi /etc/ssh/sshd_config
```

Na początek dodamy do konfiguracji wpis reprezentujący podsystem PowerShell:

```
Subsystem powershell /usr/bin/pwsh -sshs -NoLogo
```

W systemach Linux plik wykonywalny PowerShell zazwyczaj znajduje się w katalogu */usr/bin/pwsh*. Jeżeli zainstalowałeś PowerShella w innej lokalizacji, powinieneś odpowiednio zmodyfikować tę ścieżkę w pliku konfiguracyjnym.

Aby umożliwić użytkownikom zdalne logowanie za pomocą SSH, musisz skonfigurować `PasswordAuthentication` i/lub `PubkeyAuthentication`:

- Jeżeli chcesz zezwolić na uwierzytelnianie przy użyciu nazwy użytkownika i hasła, ustaw opcję `PasswordAuthentication` na `yes`:

  ```
  PasswordAuthentication yes
  ```

- Aby włączyć bardziej bezpieczną metodę uwierzytelniania, ustaw opcję `PubkeyAuthentication` na `yes`:

  ```
  PubkeyAuthentication yes
  ```

`PubkeyAuthentication`, czyli **uwierzytelnianie za pomocą klucza publicznego**, jest metodą uwierzytelniania, która opiera się na parze kluczy: *prywatnym*, który jest przechowywany na komputerze użytkownika, oraz *publicznym*, który jest zapisany na serwerze zdalnym.

Gdy użytkownik uwierzytelnia się za pomocą tego klucza prywatnego, serwer może zweryfikować tożsamość użytkownika za pomocą jego klucza publicznego. Klucz publiczny może być używany tylko do weryfikacji autentyczności klucza prywatnego lub do szyfrowania danych, które może szyfrować tylko klucz prywatny.

Używanie uwierzytelniania kluczem publicznym do zdalnego dostępu nie tylko chroni przed zagrożeniami związanymi z atakami na hasła, takimi jak ataki siłowe (ang. *brute force*) czy ataki słownikowe (ang. *dictionary attacks*), ale także zapewnia dodatkową warstwę bezpieczeństwa w przypadku naruszenia bezpieczeństwa serwera. W takich scenariuszach możliwe jest tylko ujawnienie klucza publicznego, podczas gdy klucz prywatny pozostaje bezpieczny. Ponieważ sam klucz publiczny nie wystarcza do uwierzytelnienia, ta metoda zapewnia lepsze bezpieczeństwo niż tradycyjne uwierzytelnianie za pomocą nazwy użytkownika i hasła, ponieważ w razie naruszenia bezpieczeństwa serwera hasła mogą zostać wykradzione i ponownie użyte przez napastnika.

Aby dowiedzieć się, jak wygenerować taką parę kluczy za pomocą narzędzia `ssh-keygen`, powinieneś zajrzeć na stronę *https://www.ssh.com/ssh/keygen/*.

Jeżeli jesteś zainteresowany tym, jak działa uwierzytelnianie kluczem publicznym, więcej szczegółowych informacji na ten temat znajdziesz na oficjalnej stronie SSH: *https://www.ssh.com/ssh/public-key-authentication*.

Oczywiście, oba mechanizmy uwierzytelniania można skonfigurować równocześnie, jednak jeżeli korzystasz z `PubkeyAuthentication` i żaden inny użytkownik nie używa uwierzytelniania za pomocą nazwy użytkownika i hasła, zaleca się stosowanie wyłącznie `PubkeyAuthentication`.

```
PasswordAuthentication no
PubkeyAuthentication yes
```

Jeżeli chcesz dowiedzieć się więcej o różnych opcjach pliku konfiguracyjnego *sshd*, powinieneś zapoznać się z jego stroną podręcznika `man`: *https://manpages.debian.org/jessie/openssh-server/sshd_config.5.en.html*.

Podręcznik man

Man (od ang. *manual*) to podręcznik zawierający strony pomocy, które dostarczają szczegółowych informacji na temat poleceń w systemach Linux/UNIX lub plików konfiguracyjnych. Można je porównać do systemu pomocy dostępnego w PowerShellu.

Po zakończeniu modyfikacji pliku konfiguracyjnego zrestartuj usługę `ssh`:

```
> /etc/init.d/ssh restart
```

Po restarcie usługi zaktualizowana konfiguracja zostanie załadowana do pamięci i wprowadzone zmiany zostaną aktywowane.

PowerShell Remoting w systemie macOS

Procedura włączania PowerShell Remoting przez SSH do zarządzania systemami macOS jest bardzo podobna do tej stosowanej w przypadku systemów Linux — główna różnica polega na innej lokalizacji plików konfiguracyjnych.

Pierwszym krokiem jest instalacja PowerShell Core w systemach macOS, którymi chcesz zarządzać zdalnie. Więcej szczegółowych informacji na ten temat znajdziesz na stronie *https://learn.microsoft.com/pl-pl/powershell/scripting/install/installing-powershell-on-macos*.

Teraz otwórz do edycji plik konfiguracyjny *sshd_config*:

```
> vi /private/etc/ssh/sshd_config
```

Dodaj wpis dla podsystemu PowerShell:

```
Subsystem powershell /usr/local/bin/pwsh -sshs -NoLogo
```

W kolejnym kroku musisz określić, jaki rodzaj uwierzytelniania chcesz skonfigurować dla tej maszyny:

- Nazwa użytkownika i hasło:
  `PasswordAuthentication yes`
- Uwierzytelnianie kluczem publicznym:
  `PubkeyAuthentication yes`

Aby dowiedzieć się więcej o opcjach, które można skonfigurować w konfiguracji `sshd`, powinieneś zajrzeć do poprzedniej sekcji, zatytułowanej „PowerShell Remoting w systemie Linux”.

Po zapisaniu pliku konfiguracyjnego uruchom ponownie usługę `ssh`, aby załadować nową konfigurację:

```
> sudo launchctl stop com.openssh.sshd
> sudo launchctl start com.openssh.sshd
```

Po restarcie usługi zaktualizowana konfiguracja zostanie załadowana do pamięci i wprowadzone zmiany zostaną aktywowane.

PowerShell Remoting przez SSH w systemie Windows

Zarządzanie systemami Windows przez SSH jest jak najbardziej możliwe, jednak w tej książce będę korzystać z PowerShell Remoting przez WinRM we wszystkich przykładach, ponieważ jest to domyślne ustawienie w systemach Windows.

Jeżeli jednak chcesz używać PSRemoting przez SSH w systemach Windows, upewnij się, że masz zainstalowany pakiet OpenSSH i postępuj zgodnie z instrukcjami dotyczącymi konfiguracji PSRemoting przez SSH w systemie Windows. Więcej szczegółowych informacji na ten temat znajdziesz na stronach:

- *https://docs.microsoft.com/en-us/windows-server/administration/openssh/openssh_overview*,
- *https://learn.microsoft.com/pl-pl/powershell/scripting/security/remoting/ssh-remoting-in-powershell?view=powershell-7.4#install-the-ssh-service-on-a-windows-computer*.

Czy wiesz, że...

Połączenia PSRemoting za pośrednictwem SSH nie wspierają zdalnej konfiguracji punktów końcowych ani technologii **JEA** (*Just Enough Administration*).

Włączanie PowerShell Remoting

Istnieje kilka sposobów włączenia PowerShell Remoting w Twoich systemach. Jeżeli zarządzasz tylko kilkoma maszynami w swoim laboratorium, możesz włączyć tę funkcję ręcznie. Jednak w przypadku większych środowisk warto rozważyć centralne włączenie i konfigurację PSRemoting. W tej sekcji zaprezentowane zostaną obie metody. W tabeli 3.2 przedstawiono przegląd działań konfiguracyjnych przypisanych do każdej z metod.

Warto pamiętać, że polecenie `Enable-PSRemoting` jest częścią ręcznej konfiguracji. Aby skonfigurować nasłuchiwanie na protokołach HTTP i HTTPS, konieczne będą dodatkowe kroki. Przyjrzyjmy się, jakie działania należy podjąć, aby ręcznie skonfigurować PSRemoting, co może być przydatne w takich przypadkach jak scenariusze testowe.

Tabela 3.2. Różne metody włączania PowerShell Remoting

Operacja	Polecenie Enable-PSRemoting	Zasady grupy (Group Policy)	Ręczna konfiguracja
Automatyczne uruchamianie usługi WinRM	Tak	Tak	Tak
Konfiguracja odbiornika HTTP	Tak	Tak (z wyjątkiem odbiorników niestandardowych)	Tak
Konfiguracja odbiornika HTTPS	Nie	Nie	Tak
Konfiguracja punktu końcowego	Tak	Nie	Tak
Konfiguracja zapory sieciowej	Tak	Tak	Tak

Ręczne włączanie PowerShell Remoting

Jeżeli chcesz włączyć PowerShell Remoting na pojedynczej maszynie, możesz to zrobić ręcznie, używając polecenia `Enable-PSRemoting` z poziomu powłoki uruchomionej z uprawnieniami administratora:

```
> Enable-PSRemoting
WinRM has been updated to receive requests.
WinRM service type changed successfully.
WinRM service started.

WinRM has been updated for remote management.
WinRM firewall exception enabled.
Configured LocalAccountTokenFilterPolicy to grant administrative rights remotely
↪to local users.
```

W tym przykładzie polecenie zostało wykonane pomyślnie i PowerShell Remoting został włączony.

Jeżeli zastanawiasz się, jaka jest różnica między poleceniami `Enable-PSRemoting` a `winrm quickconfig`, to pamiętaj, że w praktyce jest ona niewielka. Polecenie `Enable-PSRemoting` obejmuje wszystkie działania wykonywane przez `winrm quickconfig`, ale dodatkowo wprowadza zmiany specyficzne dla środowiska Windows PowerShell, zatem mówiąc najprościej, uruchomienie polecenia `Enable-PSRemoting` jest w zupełności wystarczające i możesz spokojnie pominąć wykonanie polecenia `winrm quickconfig`.

Komunikat o błędzie Set-WSManQuickConfig

W zależności od konfiguracji sieci przy próbie ręcznego włączenia PowerShell Remoting może zostać wyświetlony komunikat o błędzie:

```
WinRM firewall exception will not work since one of the network
connection types on this machine is set to Public. Change the network
connection type to either Domain or Private and try again.
(Wyjątek zapory WinRM nie będzie działać, ponieważ jedno z połączeń sieciowych na tym
komputerze jest ustawione jako Publiczne. Zmień typ połączenia sieciowego na Domenowy
lub Prywatny i spróbuj ponownie).
```

Ten komunikat o błędzie jest generowany przez polecenie `Set-WSManQuickConfig`, które jest uruchamiane podczas włączania PowerShell Remoting.

Pojawia się on, gdy jedno z połączeń sieciowych jest ustawione jako publiczne, ponieważ domyślnie PSRemoting nie jest dozwolony w sieciach oznaczonych jako publiczne.

```
> Get-NetConnectionProfile
Name             : Network 1
InterfaceAlias   : Ethernet
InterfaceIndex   : 4
NetworkCategory  : Public
IPv4Connectivity : Internet
IPv6Connectivity : NoTraffic
```

Aby uniknąć tego błędu, masz dwie opcje:

- Możesz skonfigurować profil sieciowy jako prywatny.
- Możesz wymusić uruchomienie polecenia `Enable-PSRemoting`, aby zignorować sprawdzanie profilu sieciowego.

Jeżeli masz pewność, że profil sieciowy nie jest publiczny, a sieć jest zaufana, możesz ustawić go jako prywatny:

```
> Set-NetConnectionProfile -NetworkCategory Private
```

Jeśli nie chcesz konfigurować tej sieci jako zaufanej sieci prywatnej, możesz wymusić pominięcie sprawdzania profilu sieciowego, dodając parametr `-SkipNetworkProfileCheck`:

```
> Enable-PSRemoting -SkipNetworkProfileCheck
```

Włączenie PSRemoting na komputerach w sieci publicznej jest bardzo ryzykowne i może narazić Twój komputer na poważne niebezpieczeństwo, więc powinieneś tutaj zachować szczególną ostrożność.

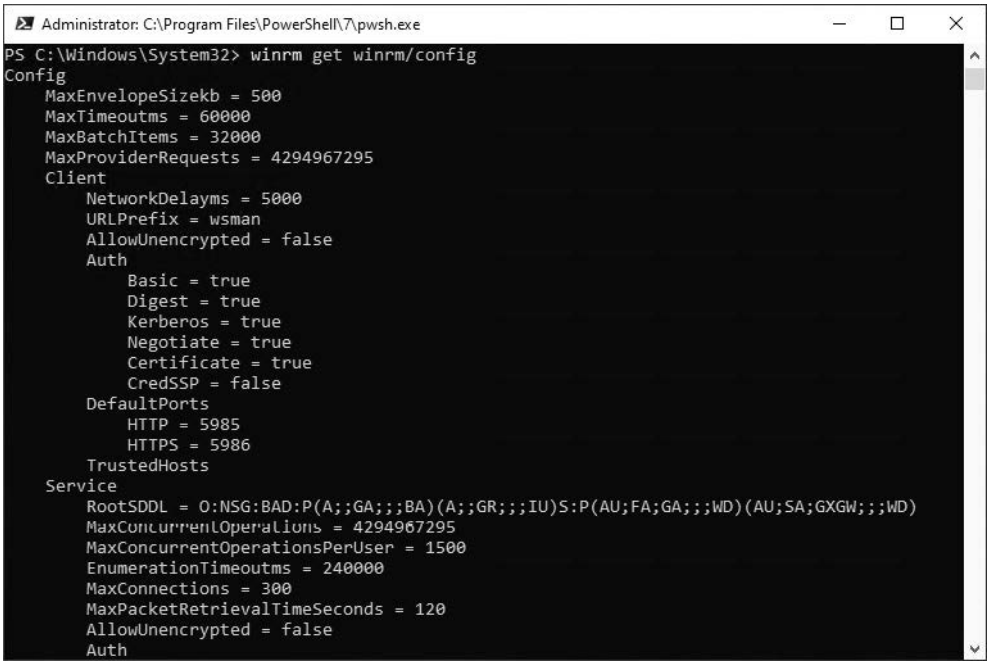
Sprawdzanie konfiguracji WinRM

Po włączeniu PowerShell Remoting i WinRM warto sprawdzić bieżącą konfigurację WinRM. Można to zrobić za pomocą polecenia `winrm get winrm/config`, tak jak pokazano na rysunku 3.2.

Wyniki działania polecenia `winrm get winrm/config` zawierają podsumowanie wszystkich ustawień konfiguracji WinRM.

Aby zmienić lokalną konfigurację WinRM, możesz użyć opcji `set`:

```
> winrm set winrm/config/service '@{AllowUnencrypted="false"}'
```



Rysunek 3.2. Weryfikacja lokalnej konfiguracji WinRM

Dostęp do określonych elementów konfiguracji możesz również uzyskać za pośrednictwem dysku `wsman:\`. Wykorzystanie tego dostawcy pozwala na dostęp i modyfikację wybranych ustawień WinRM w sposób bardziej intuicyjny i zbliżony do sposobu działania poleceń cmdlet, z dodatkową zaletą w postaci wbudowanej pomocy i obszernej dokumentacji.

Aby zmienić lokalne ustawienia WinRM, można użyć polecenia cmdlet `Set-Item` razem z dostawcą `wsman:\`. Na przykład, aby wyłączyć możliwość użycia nieszyfrowanego ruchu, możesz wykonać następujące polecenie:

```
> Set-Item wsman:\localhost\Service\AllowUnencrypted -Value $false
```

W tym przykładzie konfigurujemy usługę WinRM tak, aby blokowała połączenia nieszyfrowane. Możesz zastosować podobną składnię do skonfigurowania innych opcji WinRM — wystarczy, że określisz pełną ścieżkę do danego ustawienia w drzewie `wsman:\`, a także podasz odpowiednią opcję i wartość.

Zaufane hosty

Jeżeli łączysz się z komputerem, który nie jest przyłączony do domeny (co może być powodem, dla którego konfigurujesz go ręcznie), uwierzytelnianie Kerberos nie będzie dostępne, a zamiast tego powinieneś użyć protokołu NTLM.

W takim przypadku musisz najpierw skonfigurować zdalny komputer jako zaufanego hosta (ang. *trusted host*) na swoim lokalnym urządzeniu; inaczej połączenie się nie powiedzie.

Aby skonfigurować `TrustedHosts` dla zdalnego hosta, można użyć polecenia cmdlet `Set-Item` wraz ze ścieżką *wsman:\localhost\client\TrustedHosts*. Domyślnie ta lokalizacja jest pusta, więc musisz dodać tam adres IP lub nazwę domeny zdalnego hosta. Aby dodać nową wartość bez zastępowania już istniejących wpisów, powinieneś użyć opcji `-Concatenate`, jak pokazano w kolejnym przykładzie:

```
> Set-Item wsman:\localhost\client\TrustedHosts -Value 172.29.0.12 -Concatenate -Force
```

Spowoduje to dołączenie podanego adresu IP do istniejącej listy `TrustedHosts` (lista zaufanych hostów).

Aby sprawdzić, czy wprowadzone zmiany zostały zastosowane, można użyć polecenia `Get-Item`, za pomocą którego możesz wyświetlić bieżącą konfigurację `TrustedHosts`:

```
> Get-Item wsman:\localhost\client\TrustedHosts
WSManConfig: Microsoft.WSMan.Management\WSMan::localhost\Client
Type             Name             SourceOfValue     Value
----             ----             -------------     -----
System.String    TrustedHosts                       172.29.0.12
```

Powyższy przykład pokazuje, że host o adresie IP `172.29.0.12` został dodany do listy zaufanych hostów na komputerze lokalnym.

Regularne przeglądanie listy zaufanych hostów jest dobrą praktyką, pozwalającą na wykrycie wszelkich nieautoryzowanych modyfikacji oraz identyfikację potencjalnych prób manipulacji w systemie.

Łączenie przez HTTPS

Można również opcjonalnie skonfigurować certyfikat do szyfrowania ruchu z wykorzystaniem protokołu **HTTPS**. Dla zapewnienia większego bezpieczeństwa PowerShell Remoting zaleca się zastosowanie certyfikatu do szyfrowania ruchu za pomocą HTTPS, szczególnie w przypadkach, gdy Kerberos nie jest dostępny do uwierzytelniania serwera. Mimo że ruch PSRemoting jest szyfrowany domyślnie, takie szyfrowanie można wyłączyć, a podstawowe uwierzytelnianie można łatwo wymusić (patrz punkt „Uwierzytelnianie i kwestie bezpieczeństwa PowerShell Remoting"). Konfiguracja certyfikatu wprowadza dodatkową warstwę ochrony w Twoim środowisku.

Aby zwiększyć poziom bezpieczeństwa, warto wystawić certyfikat i włączyć **WinRM przez SSL**.

Jeżeli nie posiadasz publicznego certyfikatu SSL podpisanego przez **zaufany urząd certyfikacji** (ang. *Certificate Authority* — CA), możesz utworzyć certyfikat samopodpisany i rozpocząć konfigurację od niego. W przypadku zdalnego zarządzania w grupie roboczej możesz również wykorzystać certyfikat wydany przez **wewnętrzny urząd certyfikacji**. Taki certyfikat zapewnia dodatkowe bezpieczeństwo i zaufanie, ponieważ jest podpisany przez zaufane źródło w organizacji.

W tej sekcji omówimy jedynie sposób tworzenia i konfigurowania certyfikatu samopodpisanego. Jeżeli korzystasz z publicznego certyfikatu podpisanego przez zaufany urząd certyfikacji lub wewnętrzny urząd certyfikacji, pamiętaj, aby odpowiednio dostosować działania opisane poniżej.

Zacznijmy zatem od utworzenia certyfikatu samopodpisanego! Proces ten jest bardzo prosty, jeżeli używasz systemu Windows Server 2012 lub nowszego — w takiej sytuacji wystarczy po prostu skorzystać z polecenia `New-SelfSignedCertificate`:

```
> $Cert = New-SelfSignedCertificate -CertstoreLocation Cert:\LocalMachine\
➥My -DnsName "PSSec-PC01"
> Export-Certificate -Cert $Cert -FilePath C:\tmp\cert
```

Upewnij się, że wartość przekazana jako parametr `-DnsName` jest zgodna z nazwą hosta oraz że w Twoim serwerze DNS istnieje odpowiadający jej rekord DNS.

Następnie dodaj odbiornik dla ruchu HTTPS:

```
> New-Item -Path WSMan:\LocalHost\Listener -Transport HTTPS -Address * -
➥CertificateThumbPrint $Cert.Thumbprint –Force
```

Na koniec upewnij się, że dodałeś odpowiedni wyjątek do zapory sieciowej. Domyślny port dla WinRM przez HTTPS to `5986`:

```
> New-NetFirewallRule -DisplayName "Windows Remote Management (HTTPS-In)" -Name
➥"Windows Remote Management (HTTPS-In)" -Profile Any -LocalPort 5986 -Protocol TCP
```

Dla wyjaśnienia: pamiętaj, że zastosowanie opcji `-Profile Any` otwiera WinRM na sieci publiczne i inne sieci niezidentyfikowane. Jeżeli pracujesz w innym środowisku niż testowe, upewnij się, że używasz odpowiednich opcji profilu zapory sieciowej, takich jak `Domain`, `Private` lub `Public`.

Jeżeli chcesz mieć pewność, że używany jest wyłącznie protokół HTTPS, usuń odbiornik HTTP WinRM.

```
> Get-ChildItem WSMan:\Localhost\listener | Where -Property Keys -eq "Transport=
➥HTTP" | Remove-Item -Recurse
```

Powinieneś również usunąć wszelkie istniejące wyjątki zapory dla ruchu HTTP, które zostały skonfigurowane wcześniej. Oczywiście ten krok nie jest konieczny, jeżeli nie utworzono żadnych wyjątków.

W niektórych przypadkach może być konieczne przeniesienie odbiornika WinRM na inny port, co może być przydatne, jeżeli zapora nie pozwala na wykorzystanie portu `5986` lub jeżeli chcesz użyć niestandardowego portu z powodów bezpieczeństwa. Aby zmienić port, należy skorzystać z polecenia cmdlet `Set-Item`.

```
> Set-Item WSMan:\Localhost\listener\<NazwaOdbiornika>\port -Value <NumerPortu>
```

Zastąp opcję *<NazwaOdbiornika>* nazwą odbiornika, który chcesz edytować, a opcję *<NumerPortu>* numerem portu, który chcesz skonfigurować.

Kolejnym krokiem jest zaimportowanie naszego certyfikatu. Zanim to zrobimy, powinieneś zapamiętać, że certyfikaty wygenerowane za pomocą narzędzi takich jak `New-Self➥SignedCertificate` posiadają wbudowane ograniczenia użytkowania, które zapewniają, że są one przeznaczone wyłącznie do uwierzytelniania klienta i serwera. Jeżeli używasz certyfikatu wygenerowanego za pomocą innego narzędzia (na przykład wewnętrznego systemu PKI), powinieneś upewnić się, że ma on również te same ograniczenia użytkowania. Oprócz tego upewnij się, że Twój certyfikat root jest odpowiednio zabezpieczony, ponieważ potencjalni napastnicy mogą go użyć do fałszowania certyfikatów SSL dla zaufanych witryn internetowych.

Po uzyskaniu odpowiedniego certyfikatu skopiuj go do bezpiecznej lokalizacji na komputerze, z którego chcesz połączyć się z maszyną zdalną (w naszym przykładzie będzie to katalog *C:\tmp\cert*), a następnie zaimportuj go do lokalnego magazynu certyfikatów:

```
> Import-Certificate -Filepath "C:\tmp\cert" -CertStoreLocation "Cert:\
↪LocalMachine\Root"
```

Wybierz poświadczenia, które chcesz wykorzystać do logowania oraz nawiązywania sesji. Parametr `-UseSSL` oznacza, że połączenie będzie szyfrowane przy użyciu protokołu SSL.

```
> $cred = Get-Credential
> Enter-PSSession -ComputerName PSSec-PC01 -UseSSL -Credential $cred
```

Oczywiście nadal musisz wprowadzić poświadczenia, aby zalogować się zdalnie na komputerze. Certyfikat gwarantuje jedynie autentyczność zdalnego komputera i pomaga ustanowić szyfrowane połączenie.

Konfigurowanie PowerShell Remoting za pomocą zasad grupy (Group Policy)

Przy pracy z wieloma serwerami ręczne włączanie PSRemoting na każdym z osobna może być uciążliwe. W takim przypadku warto skorzystać z narzędzia *Zasady grupy* (*Group Policy*).

Za pomocą zasad grupy można skonfigurować wiele serwerów przy użyciu jednego obiektu zasad grupy (ang. *Group Policy Object* — GPO).

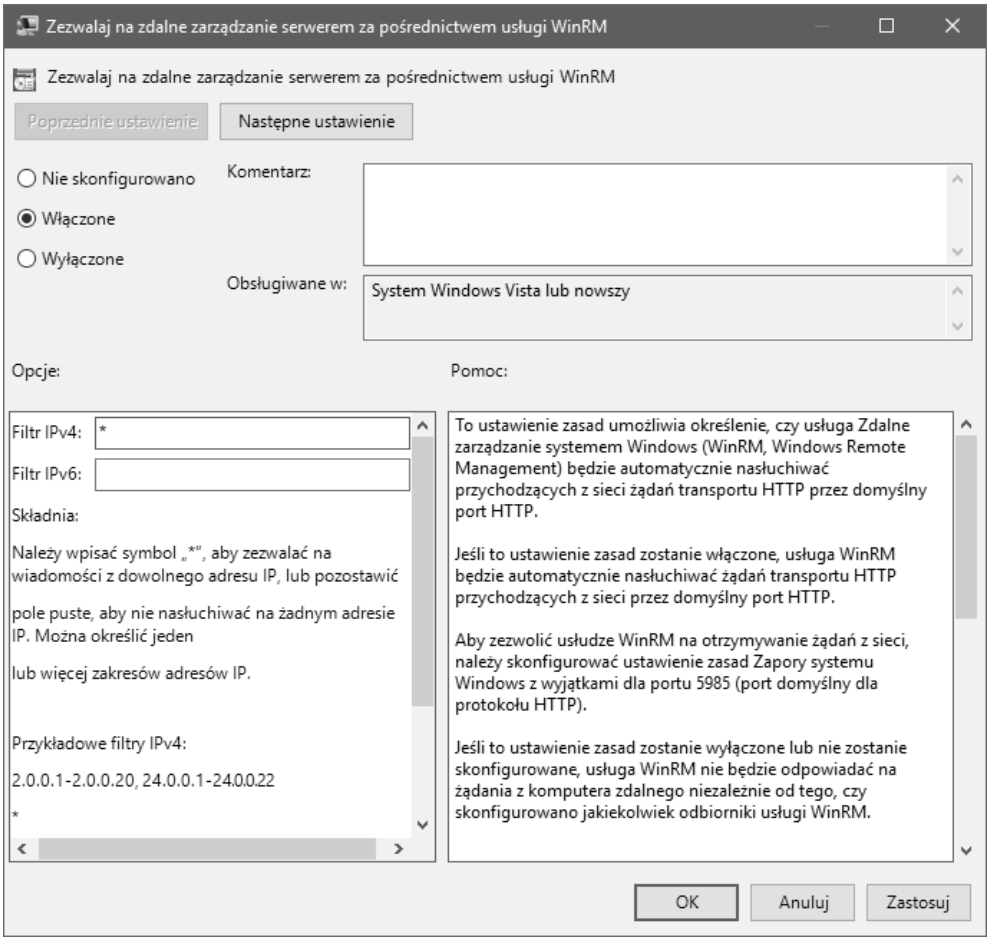
Aby rozpocząć, utwórz nowy obiekt GPO. W tym celu otwórz konsolę *Zarządzanie zasadami grupy* (ang. *Group Policy Management*), kliknij prawym przyciskiem myszy jednostkę organizacyjną (ang. *Organizational Unit* — OU), w której chcesz utworzyć nowy obiekt GPO, a następnie wybierz opcję *Utwórz obiekt GPO w tej domenie i połącz go tutaj*.

Pamiętaj, że GPO jest tylko narzędziem do konfigurowania maszyn i za jego pomocą nie możesz uruchamiać usług. W związku z tym musisz znaleźć sposób, aby ponownie uruchomić wszystkie skonfigurowane serwery lub uruchomić usługę WinRM na tych serwerach.

Jeżeli chcesz włączyć PowerShell Remoting zdalnie, możesz skorzystać ze świetnego skryptu, napisanego przez Lee Holmesa, który wykorzystuje połączenia WMI (obsługiwane przez większość systemów). Wspomniany skrypt znajdziesz pod adresem *http://www.powershellcookbook.com/recipe/SQOK/program-remotely-enable-powershell-remoting*.

Zezwalanie na połączenia za pośrednictwem WinRM

W nowo utworzonym GPO przejdź do ustawień *Konfiguracja komputera/Zasady/Szablony administracyjne/Składniki systemu Windows/Zdalne zarzadzanie systemem Windows (WinRM)/Usługa WinRM* i ustaw zasadę *Zezwalaj na zdalne zarządzanie serwerem za pośrednictwem usługi WinRM* na *Włączone*, tak jak pokazano na rysunku 3.3.



Rysunek 3.3. Włączanie zasady zezwalającej na zdalne zarządzanie serwerem przez WinRM

W tej polityce można zdefiniować filtry dla protokołów IPv4 i IPv6. Jeżeli nie korzystasz z danego protokołu (np. IPv6), pozostaw to pole puste, aby użytkownicy nie mogli łączyć się z WinRM przy jego użyciu.

Aby dopuszczać wybrane połączenia, możesz użyć symbolu wieloznacznego * lub podać konkretny adres IP albo zakres adresów IP.

Pracując z klientami i przeprowadzając testy w swoich środowiskach laboratoryjnych, zauważyłam, że najczęstszą przyczyną problemów z WinRM było użycie niepoprawnego adresu IP lub zakresu adresów IP w konfiguracji tych ustawień.

Z tego powodu obecnie używam symbolu wieloznacznego * *tylko* w połączeniu z odpowiednimi **ograniczeniami zakresu adresów IP** zdefiniowanymi w zaporze sieciowej tak, aby zapewnić bezpieczeństwo mojego środowiska. Ograniczenia zakresów IP zapory sieciowej skonfigurujemy w dalszej części tego podrozdziału (patrz punkt „Tworzenie reguł zapory sieciowej").

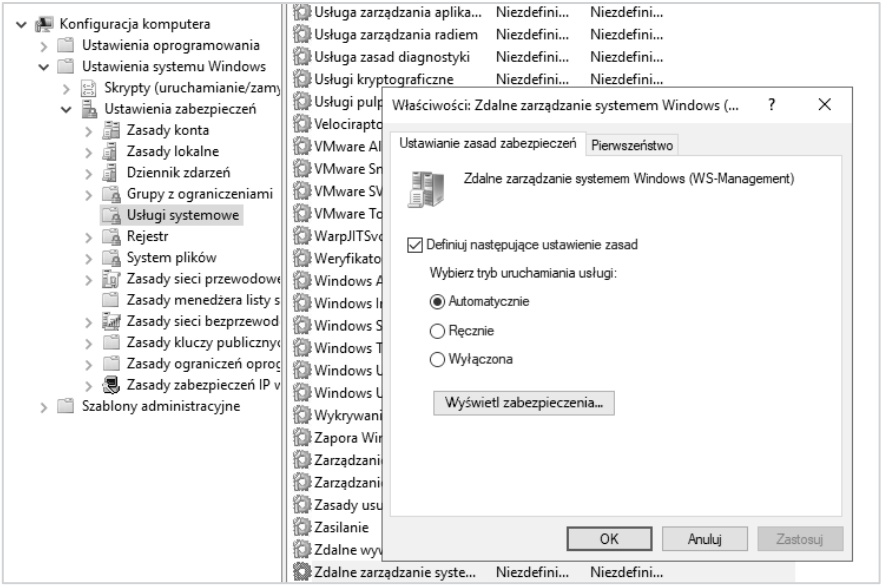
Uwaga

Konfiguracji z symbolem wieloznacznym * należy używać wyłącznie w sytuacji, kiedy planujesz ograniczenie możliwości łączenia się ze zdalnymi adresami IP za pomocą reguł zapory sieciowej.

Konfigurowanie usługi WinRM do automatycznego uruchamiania

Aby skonfigurować usługę WinRM tak, aby uruchamiała się automatycznie przy starcie systemu, powinieneś wykonać następujące operacje:

1. W edytorze GPO przejdź do opcji *Konfiguracja komputera/Zasady/Ustawienia systemu Windows/Ustawienia zabezpieczeń/Usługi systemowe*.
2. Wybierz zasadę *Zdalne zarządzanie systemem Windows (WS-Management)*.
3. Na ekranie pojawi się okno ustawień. Zaznacz opcję *Zdefiniuj to ustawienie zasad* i ustaw tryb uruchamiania usługi na *Automatyczny*, tak jak pokazano na rysunku 3.4.



Rysunek 3.4. Konfigurowanie usługi Zdalne zarządzanie systemem Windows do automatycznego uruchamiania

4. Potwierdź konfigurację, naciskając przycisk *OK*.

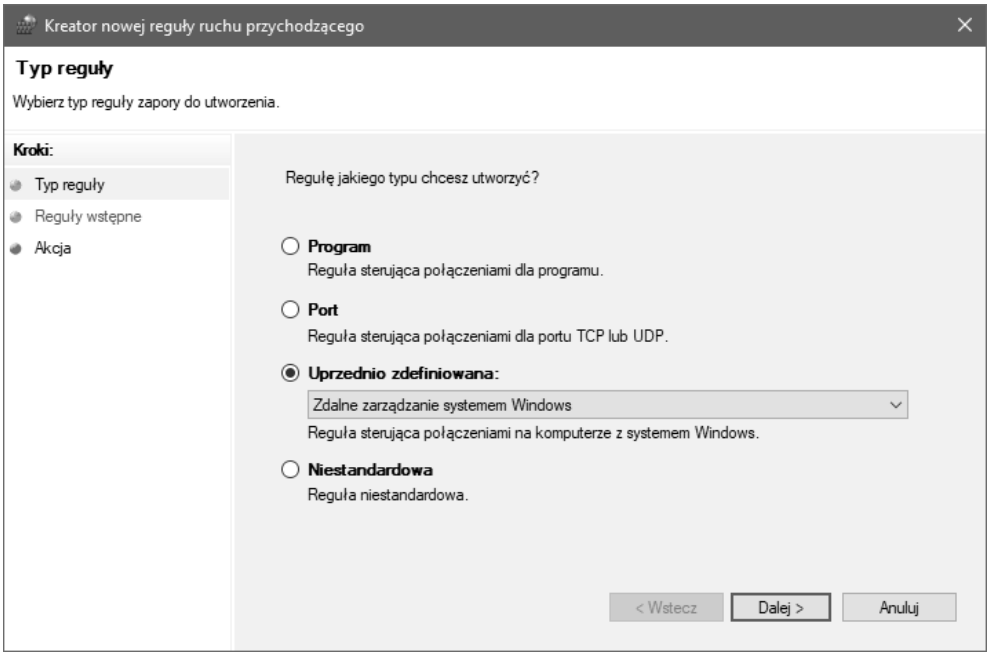
Uwaga

Pamiętaj, że to ustawienie jedynie konfiguruje usługę do automatycznego uruchamiania, co zazwyczaj dzieje się podczas startu systemu. Nie spowoduje ono jednak natychmiastowego uruchomienia usługi — użytkownik sam musi zrestartować komputer lub ręcznie uruchomić usługę, aby usługa WinRM zaczęła działać automatycznie.

Tworzenie reguł zapory sieciowej

Aby skonfigurować ustawienia zapory sieciowej, wykonaj następujące polecenia:

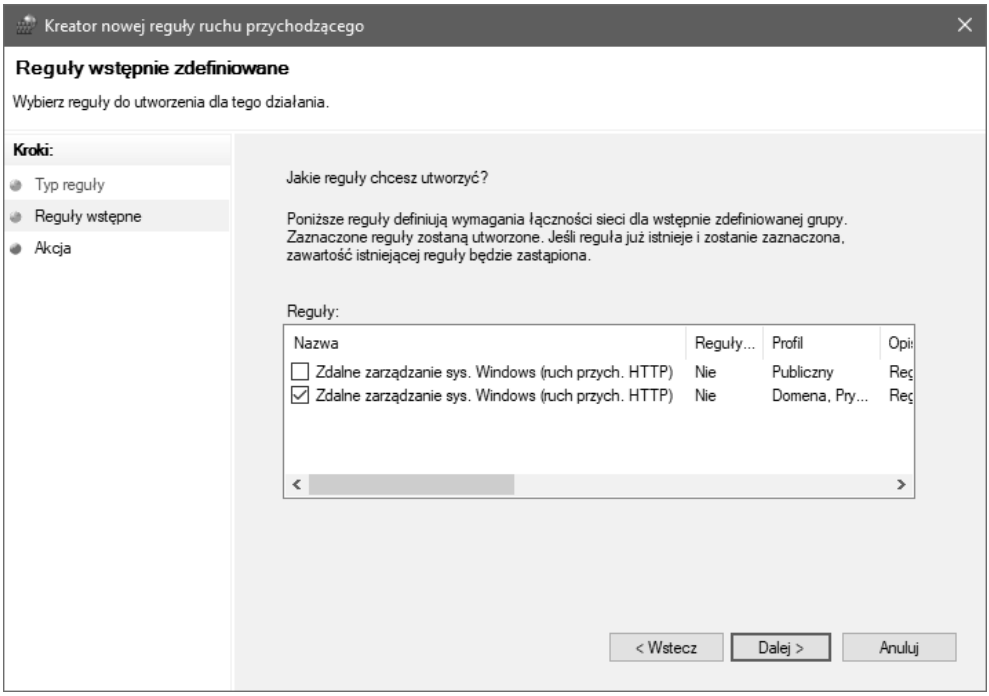
1. Przejdź do ustawień *Konfiguracja komputera/Zasady/Ustawienia systemu Windows/Ustawienia zabezpieczeń/Zapora systemu Windows z zabezpieczeniami zaawansowanymi/Zapora systemu Windows z zabezpieczeniami zaawansowanymi/Reguły przychodzące*.
2. Utwórz nową regułę przychodzącą za pomocą kreatora.
3. Zaznacz opcję *Uprzednio zdefiniowana* i z listy rozwijanej wybierz opcję *Zdalne zarządzanie systemem Windows*, tak jak pokazano na rysunku 3.5.



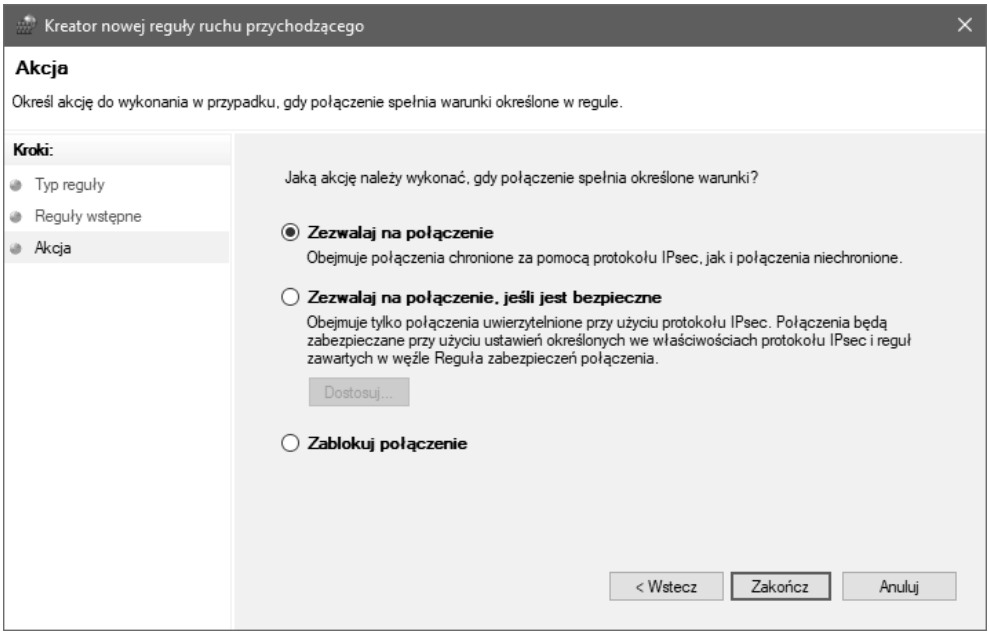
Rysunek 3.5. Tworzenie predefiniowanej reguły zapory dla usługi zdalnego zarządzania

4. Naciśnij przycisk *Dalej*, usuń publiczny profil zapory poprzez usunięcie zaznaczenia opcji *Zdalne zarządzanie sys. Windows (ruch przych. HTTP)*, tak jak pokazano na rysunku 3.6.
5. Wybierz opcję *Zezwalaj na połączenie* (patrz rysunek 3.7) i zakończ konfigurowanie reguły, naciskając przycisk *Zakończ*.

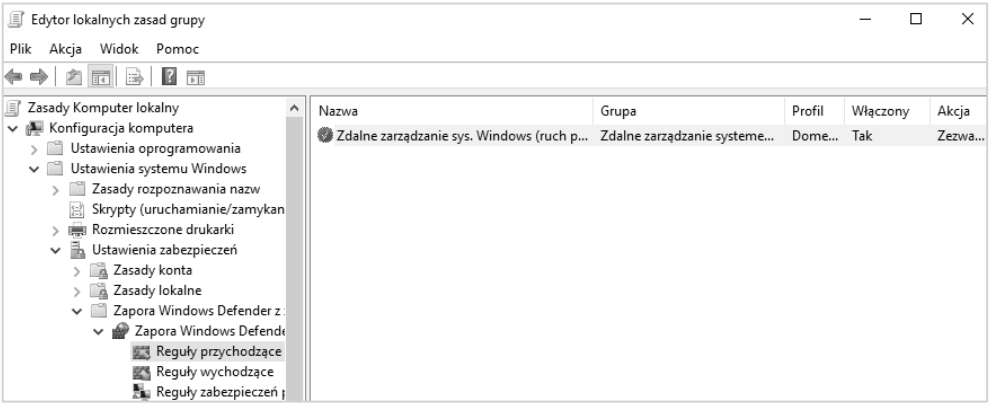
   Nowa reguła zostanie utworzona i wyświetlona w GPO, tak jak pokazano na rysunku 3.8.
6. Teraz otwórz nowo utworzoną regułę, klikając ją dwukrotnie lewym przyciskiem myszy. Na ekranie pojawi się okno *Właściwości: Zdalne zarządzanie sys. Windows (ruch przych. HTTP)*.



Rysunek 3.6. Usunięcie profilu zapory dla sieci publicznej

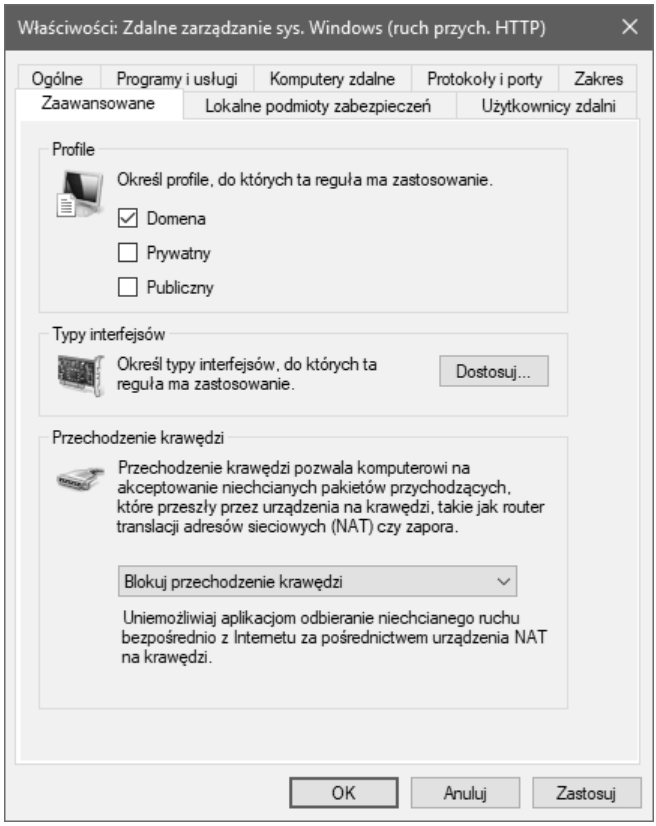


Rysunek 3.7. Konfigurowanie reguły zapory sieciowej — zezwolenie na połączenie



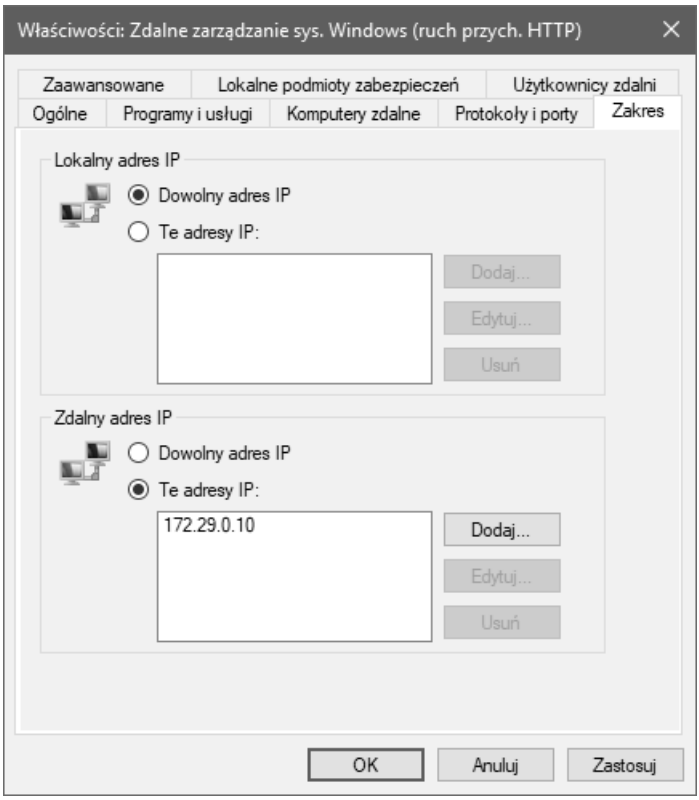
Rysunek 3.8. Wyświetlanie nowej reguły zapory sieciowej dla ruchu przychodzącego

7. Opcjonalnie: jeżeli Twoje komputery znajdują się w tej samej domenie, przejdź na kartę *Zaawansowane* i usuń zaznaczenie profilu *Prywatny*, aby upewnić się, że zdalne połączenie za pomocą WinRM będzie dozwolone tylko w profilu *Domena*, tak jak pokazano na rysunku 3.9.



Rysunek 3.9. Zezwolenie na WinRM tylko w ramach profilu sieciowego domeny

8. Następnie przejdź na kartę *Zakres* i dodaj wszystkie zdalne adresy IP, z których powinien być możliwy zdalny dostęp do komputera. Na przykład, jeżeli w swojej sieci masz podsieć zarządzania, możesz dodać adresy IP tej podsieci do listy, tak jak pokazano na rysunku 3.10.



Rysunek 3.10. Konfigurowanie zdalnych adresów IP, które mogą się łączyć

W idealnym scenariuszu zarządzanie systemami za pomocą PowerShell Remoting powinno być dozwolone tylko z poziomu bezpiecznych, odpowiednio zabezpieczonych komputerów zarządzających.

Aby stworzyć taki system, warto zastosować zasadę czystego źródła (ang. *clean source principle*) i korzystać z zalecanego modelu dostępu uprzywilejowanego, tak jak to zostało opisane w dokumentacji:

- *https://learn.microsoft.com/pl-pl/security/privileged-access-workstations/privileged-access-success-criteria#clean-source-principle*,
- *https://learn.microsoft.com/pl-pl/security/privileged-access-workstations/privileged-access-access-model*.

Punkty końcowe PowerShell (konfiguracje sesji)

W tym rozdziale spotkałeś się już co najmniej kilka razy z określeniem **punkt końcowy** (ang. *endpoint*).

Kiedy mówimy o punktach końcowych, mamy na myśli więcej niż tylko jeden komputer: PowerShell Remoting jest zaprojektowany do obsługi wielu punktów końcowych na jednym urządzeniu.

Ale czym dokładnie jest punkt końcowy?

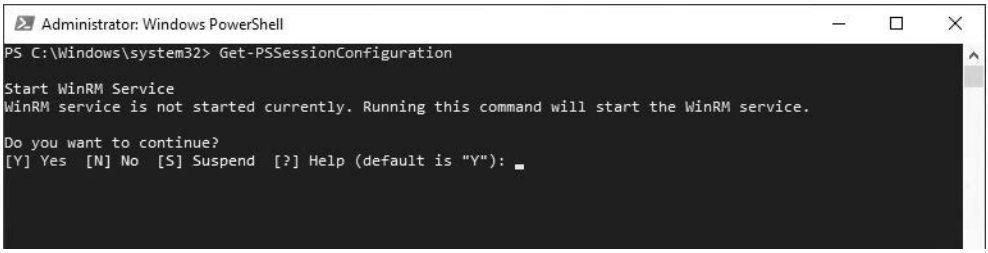
Punkt końcowy PowerShell to konfiguracja sesji, którą można ustawić w taki sposób, aby oferowała określone usługi bądź ograniczenia.

Za każdym razem, gdy używamy polecenia `Invoke-Command` lub nawiązujemy sesję PowerShell, łączymy się z punktem końcowym (znanym również jako konfiguracja sesji zdalnej).

Sesje, które mają mniej dostępnych poleceń cmdlet i funkcji niż standardowe sesje bez ograniczeń, nazywamy **ograniczonymi punktami końcowymi** (ang. *constrained endpoints*).

Przed włączeniem PowerShell Remoting na komputerze nie ma żadnego skonfigurowanego punktu końcowego PowerShell.

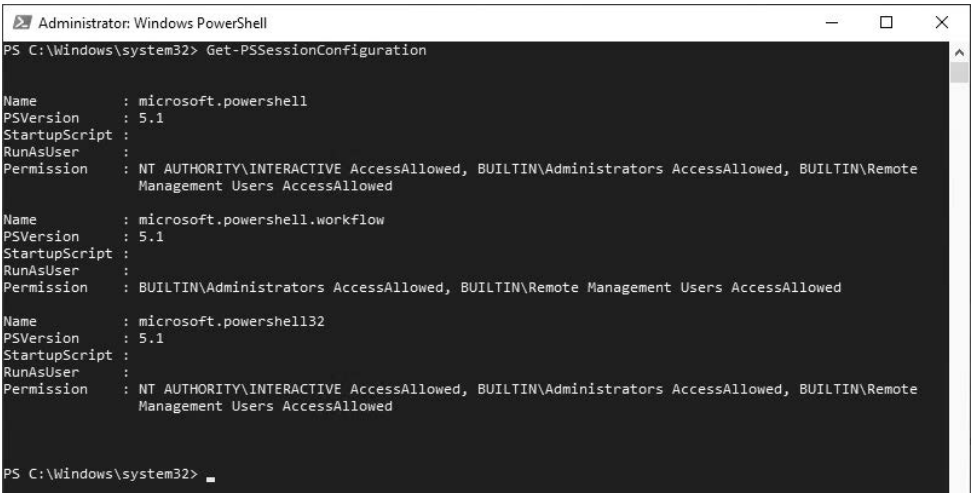
Wszystkie dostępne konfiguracje sesji można wyświetlić, uruchamiając polecenie `Get-PSSessionConfiguration`, tak jak pokazano na rysunku 3.11.



Rysunek 3.11. Jeżeli PowerShell Remoting nie jest włączony, nie ma żadnych punktów końcowych

Jeżeli PowerShell Remoting nie jest włączony na komputerze, nie będą widoczne żadne punkty końcowe. Dzieje się tak, ponieważ usługa WinRM, odpowiedzialna za PSRemoting, nie jest domyślnie aktywna. Po uruchomieniu usługi WinRM punkty końcowe są już skonfigurowane i gotowe do użycia, ale nie będą widoczne ani dostępne do połączenia, dopóki PSRemoting nie zostanie włączony.

Włączenie PowerShell Remoting za pomocą polecenia `Enable-PSRemoting`, jak to zrobiliśmy w poprzedniej sekcji, tworzy wszystkie domyślne konfiguracje sesji, które są niezbędne do połączenia się z punktem końcowym za pośrednictwem PSRemoting, tak jak pokazano na rysunku 3.12.



Rysunek 3.12. Po włączeniu PSRemoting możemy zobaczyć wszystkie wstępnie skonfigurowane punkty końcowe

Zazwyczaj w Windows PowerShell 3.0 i nowszych wersjach istnieją trzy domyślne, wstępnie skonfigurowane punkty końcowe:

- `microsoft.powershell` — standardowy punkt końcowy, używany do połączeń PSRemoting, jeśli nie określono inaczej.
- `microsoft.powershell32` — opcjonalny, 32-bitowy punkt końcowy dla 64-bitowych systemów operacyjnych.
- `microsoft.powershell.workflow` — punkt końcowy przeznaczony dla przepływów pracy PowerShell (więcej szczegółowych informacji na ten temat znajdziesz na stronie *https://docs.microsoft.com/pl-pl/system-center/sma/overview-powershell-workflows?view=sc-sma-2019*).

W *systemach serwerowych* zazwyczaj istnieje jeszcze czwarta predefiniowana konfiguracja sesji:

- `microsoft.windows.servermanagerworkflows` — ten punkt końcowy jest przeznaczony dla przepływów pracy narzędzia Menedżer Serwera (więcej szczegółowych informacji na ten temat znajdziesz na stronie *https://docs.microsoft.com/en-us/windows-server/administration/server-manager/server-manager*).

Na poszczególnych komputerach może być wyświetlana różna liczba domyślnych punktów końcowych. W poprzednim przykładzie polecenie zostało uruchomione na kliencie Windows 10, który wyświetla mniej punktów końcowych niż na przykład Windows Server 2019.

Łączenie się z określonym punktem końcowym

Domyślnie punkt końcowy `microsoft.powershell` jest używany dla wszystkich połączeń PSRemoting. Jeżeli jednak chcesz połączyć się z innym punktem końcowym, możesz to zrobić, określając punkt końcowy za pomocą parametru `-ConfigurationName`:

```
> Enter-PSSession -ComputerName PSSec-PC01 -ConfigurationName 'microsoft.
↪powershell32'
```

Nazwa konfiguracji to inaczej nazwa innego domyślnego lub niestandardowego punktu końcowego.

Tworzenie niestandardowego punktu końcowego — spojrzenie na JEA

Tworzenie niestandardowego punktu końcowego (znane również jako **Just Enough Administration** lub po prostu **JEA**) pozwala na skonfigurowanie ograniczonego środowiska administracyjnego, dedykowanego do delegowanego zarządzania. JEA pozwala na określenie zbioru dozwolonych parametrów i poleceń, które użytkownicy mogą wykonywać na wybranych komputerach. Umożliwia to nadanie użytkownikom wystarczających uprawnień do wykonywania obowiązków służbowych bez przyznawania im pełnego dostępu administracyjnego. Jest to świetny sposób na zabezpieczenie połączeń zdalnych, które oferuje kilka kluczowych funkcji:

- Ograniczenie sesji do uruchamiania jedynie wcześniej zdefiniowanych poleceń.
- Możliwość włączenia transkrypcji, która zapisuje wszystkie wykonywane polecenia w sesji.
- Określenie deskryptora zabezpieczeń (SDDL), który definiuje, kto może, a kto nie może połączyć się z danym punktem końcowym.
- Konfiguracja skryptów i modułów, które są automatycznie ładowane po nawiązaniu połączenia z tym punktem końcowym.
- Możliwość wskazania innego konta użytkownika do uruchamiania poleceń w trakcie sesji na tym punkcie końcowym.

Aby utworzyć i aktywować punkt końcowy, musisz wykonać dwa kroki:

- Utwórz plik konfiguracji sesji.
- Zarejestruj sesję jako nowy punkt końcowy.

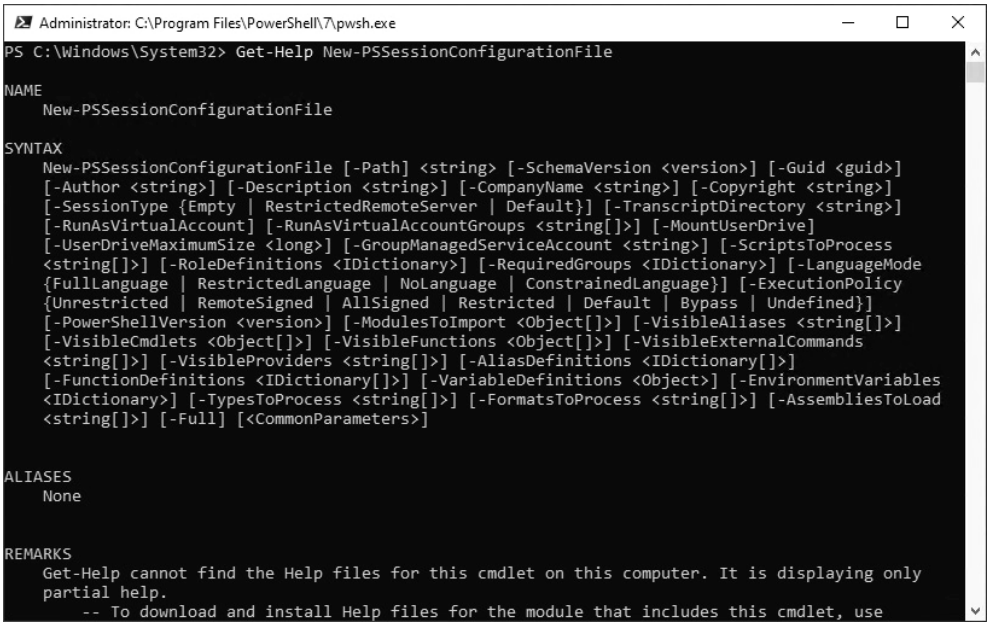
Tworzenie pliku konfiguracji sesji

Polecenie `New-PSSessionConfigurationFile` pozwala na utworzenie pustego szkieletu pliku konfiguracyjnego sesji. Wymaga ono podania ścieżki, w której plik ma zostać zapisany, dlatego parametr `-Path` jest obowiązkowy. Plik konfiguracyjny sesji powinien mieć rozszerzenie *.pssc*, więc upewnij się, że nadałeś mu odpowiednią nazwę.

```
> New-PSSessionConfigurationFile -Path <Ścieżka\NazwaPliku.pssc>
```

Więcej szczegółowych informacji na ten temat znajdziesz w oficjalnej dokumentacji na stronie *https://docs.microsoft.com/pl-pl/powershell/module/microsoft.powershell.core/new-pssessionconfigurationfile*.

Możesz albo wygenerować pusty plik konfiguracji sesji i wypełnić go później za pomocą edytora tekstu, albo użyć odpowiednich parametrów wywołania polecenia `New-PS↪SessionConfigurationFile`, aby bezpośrednio wygenerować plik ze wszystkimi niezbędnymi opcjami konfiguracyjnymi (rysunek 3.13).



Rysunek 3.13. Parametry wywołania polecenia New-PSSessionConfigurationFile

W kolejnym przykładzie utworzymy plik konfiguracji dla sesji `RestrictedRemoteServer`:

```
> New-PSSessionConfigurationFile -SessionType RestrictedRemoteServer -Path .\
↪PSSessionConfig.pssc
```

Kiedy używasz parametru `-SessionType RestrictedRemoteServer`, do sesji importowane są tylko najważniejsze polecenia, takie jak `Exit-PSSession`, `Get-Command`, `Get-FormatData`, `Get-Help`, `Measure-Object`, `Out-Default` i `Select-Object`. Jeżeli chcesz zezwolić na wykonywanie innych poleceń w takiej sesji, musisz dodać je do pliku definicji roli, który omówimy szczegółowo w rozdziale 10. „Tryby językowe sesji PowerShell i funkcjonalność JEA".

Rejestrowanie sesji jako nowego punktu końcowego

Po utworzeniu pliku konfiguracyjnego sesji należy zarejestrować go jako punkt końcowy za pomocą polecenia `Register-PSSessionConfiguration`.

Korzystając z obowiązkowego parametru `-Name`, upewnij się, że podajesz tylko nazwę pliku konfiguracji sesji, bez uwzględniania rozszerzenia nazwy pliku:

```
> Register-PSSessionConfiguration -Name PSSessionConfig
WARNING: Register-PSSessionConfiguration may need to restart the WinRM service if
↪a configuration using this name has recently been unregistered, certain system data
↪structures may still be cached. In that case, a restart of WinRM may be required.
All WinRM sessions connected to Windows PowerShell session configurations, such
↪as Microsoft.PowerShell and session configurations that are created with the
↪Register-PSSessionConfiguration cmdlet, are disconnected.

    WSManConfig: Microsoft.WSMan.Management\WSMan::localhost\Plugin
```

```
Type          Keys                                Name
----          ----                                ----
Container     {Name=PSSessionConfig}              PSSessionConfig
```

Konfiguracja sesji zostanie zarejestrowana i zostanie utworzony nowy punkt końcowy. Czasami po zarejestrowaniu punktu końcowego może być konieczne ponowne uruchomienie usługi WinRM:

```
> Get-PSSessionConfiguration -Name PSSessionConfig

Name             : PSSessionConfig
PSVersion        : 5.1
StartupScript    :
RunAsUser        :
Permission       : NT AUTHORITY\INTERACTIVE AccessAllowed,
                   BUILTIN\Administrators AccessAllowed,
                   BUILTIN\Remote Management Users AccessAllowed
```

Za pomocą polecenia `Get-PSSessionConfiguration` można sprawdzić, czy punkt końcowy został utworzony. Jeżeli podasz nazwę punktu końcowego za pomocą parametru `-Name`, jak w poprzednim przykładzie, otrzymasz tylko informacje dotyczące określonego punktu końcowego.

Więcej szczegółowych informacji na temat konfiguracji sesji i parametrów rejestracji znajdziesz w rozdziale 10., „Tryby językowe sesji PowerShell i funkcjonalność JEA".

Uwierzytelnianie i kwestie bezpieczeństwa PowerShell Remoting

Cały ruch PSRemoting jest domyślnie szyfrowany, niezależnie od tego, czy połączenie zostało zainicjowane przez HTTP, czy HTTPS. Podstawowym używanym protokołem jest WS-Man, który został zaprojektowany tak, aby umożliwić jego szerokie zastosowanie. PowerShell Remoting wykorzystuje protokoły uwierzytelniania, takie jak Kerberos lub NTLM, do weryfikacji sesji, a protokół SSL/TLS jest używany do szyfrowania ruchu, niezależnie od tego, czy połączenie zostało zainicjowane przez HTTP, czy HTTPS.

Niemniej jednak bezpieczeństwo PowerShell Remoting, podobnie jak w przypadku każdego innego systemu, zależy od jakości zabezpieczeń samego komputera. Jeżeli administrator nie zabezpieczy wystarczająco dobrze swoich poświadczeń logowania, napastnicy mogą je przejąć i wykorzystać do swoich niecnych celów, dlatego kluczową sprawą jest skupienie się na odpowiednim zabezpieczeniu infrastruktury i najcenniejszych tożsamości.

Więcej informacji na temat bezpieczeństwa Active Directory i zabezpieczaniu poświadczeń możesz znaleźć w rozdziale 6. „Active Directory — ataki i przeciwdziałanie", a więcej informacji na temat środków zaradczych można znaleźć w części III „Zabezpieczanie PowerShella — skuteczne metody zabezpieczeń".

Warto pamiętać, że włączenie PowerShell Remoting nie gwarantuje automatycznie zapewnienia bezpieczeństwa. Tak jak w przypadku każdej technologii zdalnego zarządzania, kluczowym czynnikiem jest odpowiednie „utwardzenie" systemów i wdrożenie odpowiednich środków ochrony przed potencjalnymi zagrożeniami. Dotyczy to nie tylko

PSRemoting, ale także innych narzędzi do zdalnego zarządzania, takich jak RDP. Przeznaczając odpowiednie siły i środki na zabezpieczenie systemów oraz swojego środowiska, możesz zminimalizować ryzyko i skuteczniej chronić zasoby organizacji.

Na początek przyjrzyjmy się zatem, jak działa uwierzytelnianie w PowerShell Remoting.

Uwierzytelnianie

Domyślnie WinRM wykorzystuje protokół **Kerberos** do uwierzytelniania, a jeżeli nie jest on dostępny, przechodzi na **NTLM**.

W przypadku pracy w domenie Kerberos jest preferowanym protokołem uwierzytelniania. Aby korzystać z protokołu Kerberos w PowerShell Remoting, zarówno klient, jak i serwer muszą być częścią tej samej domeny, a nazwy DNS muszą być poprawnie skonfigurowane i dostępne. Dodatkowo, z perspektywy protokołu Kerberos, serwer musi być zarejestrowany w usłudze Active Directory.

Ogólnie rzecz biorąc, możesz określić, który protokół ma być używany podczas łączenia się ze zdalnym komputerem.

```
> Enter-PSSession -ComputerName PSSEC-PC01 -Authentication Kerberos
```

Jeżeli podczas nawiązywania sesji PSRemoting nie zostanie podany parametr -Authentication, domyślnie używana jest wartość Default, która odpowiada ustawieniu Negotiate. Oznacza to, że klient i serwer będą negocjowały najlepszy dostępny protokół uwierzytelniania, biorąc pod uwagę protokoły wspierane przez oba systemy. Zwykle preferowanym protokołem uwierzytelniania jest Kerberos, jednak jeżeli nie jest on dostępny, system automatycznie przechodzi na NTLM. Więcej szczegółowych informacji na temat `Negotiate` możesz znaleźć w dokumentacji Microsoftu na stronie *https://learn.microsoft.com/en-us/windows/win32/secauthn/microsoft-negotiate*.

W jakich sytuacjach może dojść do przełączenia uwierzytelniania na protokół NTLM?

PSRemoting został stworzony z myślą o integracji z Active Directory, co sprawia, że Kerberos jest preferowanym protokołem uwierzytelniania. Niemniej w pewnych sytuacjach uwierzytelnianie Kerberos nie jest możliwe i wtedy wykorzystywany jest protokół NTLM.

Kerberos:

- Komputery są dołączone do tej samej domeny lub znajdują się w domenach zaufanych.
- Klient jest w stanie zidentyfikować nazwę hosta lub adres IP serwera.
- Serwer ma prawidłowo zarejestrowaną w Active Directory nazwę **SPN** (ang. *Service Principal Name*), zgodną z celem połączenia.

NTLM:

- Jest powszechnie stosowany do łączenia się ze stacjami roboczymi, które nie są członkami domeny.
- Jeśli zamiast nazw DNS używane są adresy IP.

Na przykład, aby połączyć się z komputerem `PSSec-PC01` z wykorzystaniem protokołu Kerberos, możesz użyć następującego polecenia:

```
> Enter-PSSession -ComputerName PSSec-PC01
```

Jeżeli poświadczenia nie zostały jawnie określone, bieżący użytkownik ma uprawnienia dostępu do komputera zdalnego i komputer zdalny jest poprawnie skonfigurowany do akceptowania uwierzytelniania Kerberos, połączenie zostanie nawiązane automatycznie bez konieczności podawania jakichkolwiek jawnych poświadczeń. Jedną z zalet stosowania uwierzytelniania Kerberos jest więc to, że proces uwierzytelniania odbywa się w tle i jest niewidoczny dla użytkownika.

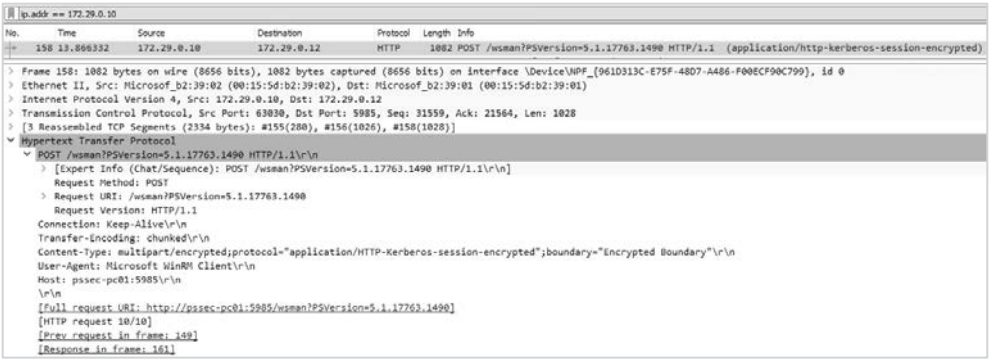
Jeżeli bieżący użytkownik nie ma uprawnień dostępu do komputera zdalnego, możemy za pomocą parametru `-Credential` jawnie określić, które poświadczenia powinny zostać użyte. Dla ułatwienia testowania możesz użyć polecenia `Get-Credential`, aby poprosić użytkownika o poświadczenia i przechowywać je w bezpiecznym ciągu `$cred`, tak jak to zostało pokazane w kolejnym przykładzie.

```
$cred = Get-Credential -Credential "PSSEC\Administrator"
```

Następnie należy nawiązać połączenie z wykorzystaniem protokołu Kerberos:

```
Enter-PSSession -ComputerName PSSEC-PC01 -Credential $cred
```

Jeżeli przechwycisz ruch sieciowy za pomocą Wiresharka, zauważysz, że w ramkach WinRM nagłówek `content-type` wskazuje, że do uwierzytelniania wykorzystywany jest protokół Kerberos. Choć sam ruch Kerberos może nie być widoczny w pakiecie HTTP, można potwierdzić jego użycie do uwierzytelniania, sprawdzając nagłówki w ruchu WinRM. Co więcej, możesz się przekonać, że cała sesja HTTP jest zaszyfrowana, co zapewnia dodatkowy poziom bezpieczeństwa (rysunek 3.14).



Rysunek 3.14. Ruch HTTP WinRM przechwycony za pomocą Wiresharka

Jak widać, sesja z komputerem `PSSec-PC01` została ustanowiona na porcie `5985` (WinRM przez HTTP) przy użyciu PowerShella w wersji 5.1.17763.1490. Żądanie zostało wysłane za pośrednictwem WS-Man.

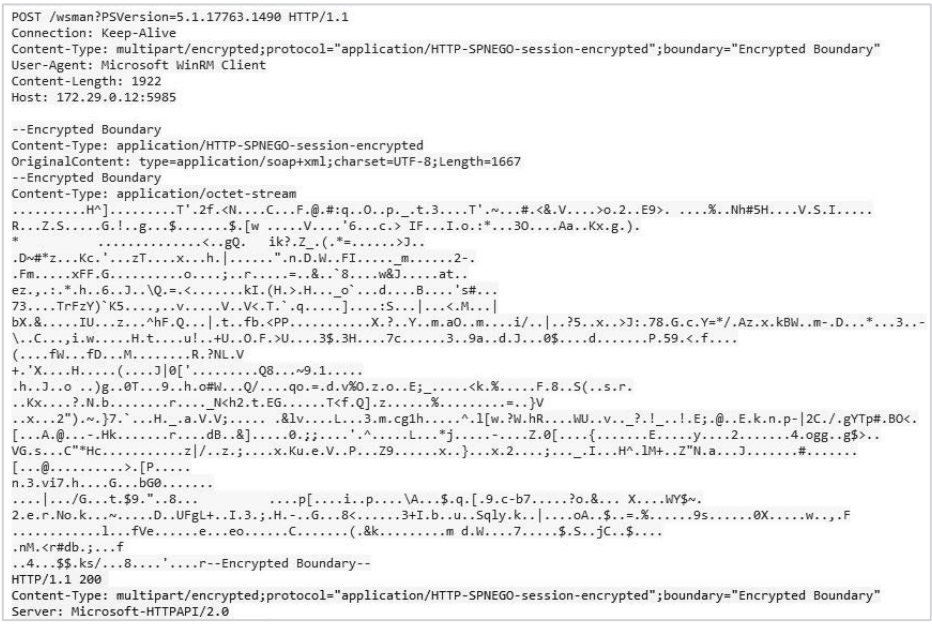
Po zakończeniu procesu uwierzytelniania WinRM nie spoczywa na laurach i kontynuuje szyfrowanie całej komunikacji, dbając o bezpieczeństwo danych przesyłanych między klientem a serwerem. Gdy nawiązujesz połączenie przez HTTPS, protokół TLS przejmuje

stery, negocjując odpowiednią metodę szyfrowania dla przesyłanych informacji. W przypadku połączenia HTTP to protokół uwierzytelniania użyty na początku sesji decyduje o metodzie szyfrowania, która później chroni dane na poziomie wiadomości.

Poziomy szyfrowania zapewniane przez poszczególne protokoły uwierzytelniania są następujące:

- **Uwierzytelnianie typu Basic** — brak szyfrowania.
- **Uwierzytelnianie NTLM** — szyfrowanie za pomocą algorytmu RC4 z kluczem 128-bitowym.
- **Uwierzytelnianie Kerberos** — algorytm szyfrowania jest określony na podstawie zawartości pola `etype` w bilecie TGS (ang. *Ticket Granting Service*).
  We współczesnych systemach jest to zazwyczaj algorytm AES-256.
- **Uwierzytelnianie CredSSP** — w tym przypadku używany jest zestaw szyfrów TLS, który został wynegocjowany w uzgodnieniu.

Warto zauważyć, że choć do samego połączenia używany jest protokół HTTP, to przesyłana zawartość jest szyfrowana odpowiednim algorytmem, który wynika z początkowo zastosowanego protokołu uwierzytelniania. Istnieje powszechne błędne przekonanie, że połączenia PowerShell Remoting przez WinRM przy użyciu protokołu HTTP nie są szyfrowane. Jak pokazano na rysunku 3.15, nie jest to jednak prawdą.



```
POST /wsman?PSVersion=5.1.17763.1490 HTTP/1.1
Connection: Keep-Alive
Content-Type: multipart/encrypted;protocol="application/HTTP-SPNEGO-session-encrypted";boundary="Encrypted Boundary"
User-Agent: Microsoft WinRM Client
Content-Length: 1922
Host: 172.29.0.12:5985

--Encrypted Boundary
Content-Type: application/HTTP-SPNEGO-session-encrypted
OriginalContent: type=application/soap+xml;charset=UTF-8;Length=1667
--Encrypted Boundary
Content-Type: application/octet-stream
.........H^].........T'.2f.<N....C...F.@.#:q..O..p._.t.3....T'.~...#.<&.V....>o.2..E9>. ....%..Nh#5H....V.S.I.....
R...Z.S.....G.!..g...$.......$.[w .....V....'6...c.> IF...I.o.:*...3O....Aa..Kx.g.).
*          ..............<..gQ.   ik?.Z_.(.*=......>J..
.D~#*z...Kc.'...zT....x...h.|......".n.D.W..FI....._m......2-.
.Fm.....xFF.G..........o....;..r.....=..&..`8....w&J.....at..
ez.,.:.*.h..6..J..\Q.=.<.......kI.(H.>.H..._o`...d....B....'s#...
73....TrFzY)`K5....,..v.....V..V<.T.`.q.....]....:S...|...<.M...|
bX.&.....IU...z...^hF.Q...|.t..fb.<PP...........X.?..Y..m.aO..m....i/..|..?5..x..>J:.78.G.c.Y=*/.Az.x.kBW..m-.D...*...3..-
\..C...,i.w.....H.t....u!..+U..O.F.>U....3$.3H....7c......3..9a..d.J...0$....d.......P.59.<.f....
(....fW...fD...M........R.?NL.V
+.'X....H.....(....J|0['.........Q8...~9.1.....
.h..J..o ..)g..0T...9..h.o#W...Q/....qo.=.d.v%O.z.o..E;_.....<k.%.....F.8..S(..s.r.
..Kx....?.N.b........r...._N<h2.t.EG......T<f.Q].z......%.........=..}V
..x...2").~.}7.`...H._.a.V.V;..... .&lv....L...3.m.cg1h.....^.l[w.?W.hR....WU..v..._?.!_..!.E;.@..E.k.n.p-|2C./.gYTp#.BO<.
[...A.@...-.Hk.......r....dB..&].....0.;;....'.^.....L...*j.....-....Z.0[....{.......E.....y....2.......4.ogg..g$>..
VG.s...C"*Hc..........z|/..z.;....x.Ku.e.V..P...Z9......x..}...x.2....;..._.I...H^.lM+..z"N.a...J.......#.......
[...@.........>.[P.....
n.3.vi7.h....G...bG0.......
....|.../G...t.$9."..8...          ....p[....i..p....\A...$.q.[.9.c-b7.....?o.&... X....WY$~.
2.e.r.No.k...~.....D..UFgL+..I.3.;.H.-..G...8<......3+I.b..u..Sqly.k..|....oA..$..=.%......9s......0X.....w..,.F
............l...fVe......e...eo......C........(.&k.........m d.W....7.....$.S..jC..$....
.nM.<r#db.;...f
..4...$$.ks/...8....'....r--Encrypted Boundary--
HTTP/1.1 200
Content-Type: multipart/encrypted;protocol="application/HTTP-SPNEGO-session-encrypted";boundary="Encrypted Boundary"
Server: Microsoft-HTTPAPI/2.0
```

Rysunek 3.15. Strumień TCP Kerberos przechwycony za pomocą Wiresharka

Jeżeli nazwy DNS nie działają i oba hosty nie należą do tej samej domeny, protokół NTLM zostanie użyty jako opcja awaryjna.

Kiedy łączysz się ze zdalnym komputerem w tej samej domenie z poprawnie działającymi nazwami DNS, protokół NTLM nadal zostanie użyty do połączenia, jeżeli zamiast nazwy hosta podasz jego adres IP.

```
Enter-PSSession -ComputerName 172.29.0.12 -Credential $cred
```

Przechwycenie takiego połączenia za pomocą Wiresharka pokazuje, że do uwierzytelnienia został użyty protokół NTLM i że cały ruch jest również szyfrowany, tak jak pokazano na rysunku 3.16.

Rysunek 3.16. Połączenie z wykorzystaniem protokołu NTLM, przechwycone przez Wiresharka

Podobnie jak w przypadku połączenia Kerberos, można tutaj zauważyć, że połączenie jest nawiązywane z hostem o adresie IP `172.29.0.12` przy użyciu protokołu WinRM przez HTTP (port `5985`). Tym razem jednak do negocjowania sesji zamiast protokołu Kerberos używany jest protokół NTLM. Kiedy korzystasz z NTLM, można nawet przechwycić nazwę hosta, nazwę użytkownika, nazwę domeny i wyzwanie (ang. *challenge*), które zostało użyte do uwierzytelniania.

Po zagłębieniu się w analizę strumienia TCP staje się oczywiste, że cała komunikacja jest szyfrowana, nawet gdy używany jest protokół NTLM, jak pokazano na rysunku 3.17.

```
POST /wsman?PSVersion=5.1.17763.1490 HTTP/1.1
Connection: Keep-Alive
Content-Type: multipart/encrypted;protocol="application/HTTP-Kerberos-session-encrypted";boundary="Encrypted Boundary"
User-Agent: Microsoft WinRM Client
Content-Length: 2016
Host: pssec-pc01:5985

--Encrypted Boundary
Content-Type: application/HTTP-Kerberos-session-encrypted
OriginalContent: type=application/soap+xml;charset=UTF-8;Length=1715
--Encrypted Boundary
Content-Type: application/octet-stream
<................BpG...}e@q...K.".h...w.U:....N]....b.oy...B.......;./&.'.L.f ......E!...f.....k.....7.../.
.}Gs......V..U.3.....R,r..*....%.+.6...^.t.@Ui./Ic...y...._.em.nt.......u...j..9......kw.w.vQ>1........m.
.F..$      .mp.sy...........d..R..4..;.4.....%|.D....I
.....m..7.g..[........0..f.S...2..6.!.`.......%%}y....^v
o.Y......Z..*../..W.c........]v....n8..h.&.;.+1..Z...(D..g.R^...G.....|.{....{.. ..z...d.*.2.n..A.......
[{Uv..:.>.......I.q..[.>cL.#..t...qp..}..Rd....6"..j......
..J.:}I.CG..x0...Z.T..o..wG.o;.....|/...64.c0.SP.........b...+5.T.L*)....sJ..^./s+....I.V.$..
.....!....+XU..#.0.C......f.c.#S%<.;..u..3...
.......k+...V=.1.Z..v...B.z.......n.o..G.7`.M.K.o.....V$.1aXk...
.sCa.Y;.c...x.......9....".XL...^.kJ..g...$..sc.{X.[..OU..P^l.Hi>.....Q..2.Z..).J../...y{fS}q..D+..z.n..
vv..k.nRZ.*G.9
.5....J....4.&.).....|..v..........
N..CBR..=JG.=._.f....{@.j........g...^...A/r..*..7....X.M.           .U......fQ.J.[.t.......c..XM7.....Y.I...j.1....
[YZs.-..A.(..Z.P....; .$....#S......k4.,k...
%.a.R../.|.....G......T...'....G"(/._:..L..Zz<:..;A..Z4....&.......O.^..=W{r....6H.yH!..,.
$.y#......:.......A......^.@`.P...-`.P..Z)...
.,.i/.w.....}.l...b...8.]...1..M..*...-.....).&..0....*.
..p...x..T.oJ-Wl..K..8.<..........1          \.F.nv...49.7.T..;..f...y....U...t............+.
6.6....p6..........e..=Fw..m..e0I.VT.F
.F.i.U.|n..P6..).*h...x#.....1QC...'.....*..D..uu..._....J4` ...M.........G..[.O..W..M.h.\;[.+++Q./....Bm...V.gp.l.!
..Z..y.e..8...}}.._._"...O.F.e.
.r.&.e.h?.u....?.y]k.z'.m!
.`......G(....D.R...v.-..y...b&y....X(..x.r.S>vGv]..G.............'....h.or.*.a...T.Y..G...^&.@....M.N..@....[... .q..<...*.
%p..?..U Y.4....v....{....i..r.mtJ...H Q..<.g*t..H.c=0.....,...AN.1....m.q....j.q
...=.aE..O..js.K.......\.e....c..&..b..Mi4....JS..1..d.*...)....~\6..<.9..kY.Fj=$.z.Z?..a--Encrypted Boundary--
HTTP/1.1 200
```

Rysunek 3.17. Strumień TCP NTLM przechwycony za pomocą Wiresharka

Kiedy korzystasz z uwierzytelniania NTLM, powinieneś pamiętać, że PowerShell Remoting będzie działać tylko wtedy, gdy zdalny host zostanie dodany do listy `TrustedHosts`.

Podczas korzystania z uwierzytelniania NTLM powinieneś pamiętać o ograniczeniach związanych z listą zaufanych hostów (`TrustedHosts`). Dodanie zdalnego hosta do listy `TrustedHosts` może pomóc w rozwiązaniu niektórych problemów, ale z całą pewnością nie jest to niezawodny sposób na zapewnienie bezpiecznej komunikacji. NTLM nie gwarantuje, że nawiązujesz połączenie z odpowiednim, zamierzonym hostem, co sprawia, że poleganie na liście `TrustedHosts` może być ryzykowne. Warto zauważyć, że główną słabością protokołu NTLM jest brak możliwości zweryfikowania tożsamości zdalnego hosta, co oznacza, że nawet dodanie hosta do listy `TrustedHosts` nie czyni połączeń NTLM bardziej wiarygodnymi.

Jeżeli host nie znajduje się na liście zaufanych hostów, a poświadczenia logowania nie zostały podane jawnie (np. przez użycie parametru `-Credential $cred`), próba nawiązania zdalnej sesji lub zdalnego uruchomienia polecenia zakończy się niepowodzeniem i pojawi się komunikat o błędzie.

```
> Enter-PSSession -ComputerName 172.29.0.10
Enter-PSSession : Connecting to remote server 172.29.0.10 failed with the following
↪error message : The WinRM client cannot process the request. If the authentication
↪scheme is different from Kerberos, or if the client computer is not joined to
↪a domain, then HTTPS transport must be used or the destination machine must be added
↪to the TrustedHosts configuration setting. Use winrm.cmd to configure TrustedHosts.
Note that computers in the TrustedHosts list might not be authenticated. You can get
↪more information about that by running the following command: winrm help config.
For more information, see the about_Remote_Troubleshooting Help topic.
At line:1 char:1
+ Enter-PSSession -ComputerName 172.29.0.10
+ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
+ CategoryInfo     : InvalidArgument: (172.29.0.10:String) [Enter-PSSession],
↪PSRemotingTransportException + FullyQualifiedErrorId : CreateRemoteRunspaceFailed
```

Kerberos i NTLM nie są jedynymi dostępnymi protokołami uwierzytelniania, ale w porównaniu z innymi wyróżniają się najwyższym poziomem bezpieczeństwa. W kolejnej sekcji dowiesz się, jakie inne metody uwierzytelniania są dostępne i jak można je wymusić.

Protokoły uwierzytelniania

Oczywiście, istnieje także możliwość skonfigurowania preferowanej metody uwierzytelniania poprzez ustawienie parametru `-Authentication`.

> **Protokoły uwierzytelniania**
>
> Jeśli tylko jest to możliwe, powinieneś zawsze wybierać uwierzytelnianie Kerberos, ponieważ protokół ten gwarantuje najlepsze zabezpieczenia i najbardziej zaawansowane funkcje ochrony.
>
> Więcej szczegółowych informacji na temat uwierzytelniania i działania protokołów Kerberos i NTLM znajdziesz w rozdziale 6. „Active Directory — ataki i przeciwdziałanie".

Poniżej przedstawiono wszystkie akceptowane wartości parametru `-Authentication`:

- `Default` — wartość domyślna, w tym przypadku zostanie użyta opcja `Negotiate`.
- `Basic` — uwierzytelnianie typu `Basic` (podstawowe) jest wykorzystywane w protokole HTTP do weryfikacji użytkowników, ale samo w sobie nie zapewnia odpowiedniego poziomu bezpieczeństwa — ani dla danych przesyłanych w postaci zwykłego tekstu przez sieć, ani dla poświadczeń logowania. Jeśli jednak jest używane w połączeniu z protokołem TLS, może stanowić względnie bezpieczny mechanizm, który jest szeroko stosowany na wielu stronach internetowych.

  Ponieważ dane uwierzytelniające są jedynie kodowane w formacie Base64 (a nie szyfrowane), kodowanie to można łatwo zdekodować, co umożliwia dostęp do poświadczeń w postaci zwykłego tekstu.

  Bez szyfrowania za pomocą **SSL/TLS** ten sposób uwierzytelniania nie zapewnia poufności przekazywanych poświadczeń logowania.
- `Credssp` — gdy używasz uwierzytelniania `CredSSP`, PowerShell przesyła poświadczenia użytkownika z klienta do zdalnego serwera w celu uwierzytelnienia. Jest to szczególnie przydatne, gdy zdalna sesja musi uwierzytelnić użytkownika w kolejnych węzłach sieci. Po uwierzytelnieniu poświadczenia są przekazywane między klientem a serwerem w zaszyfrowanym formacie zapewniającym bezpieczeństwo.

  Pamiętaj, że podczas korzystania z `CredSSP` PowerShell przekazuje pełne poświadczenia użytkownika do zdalnego serwera. Oznacza to, że jeżeli połączysz się z zagrożonym komputerem, atakujący może próbować wyodrębnić Twoje poświadczenia bezpośrednio z pamięci. Warto zauważyć, że uwierzytelnianie `CredSSP` jest domyślną metodą uwierzytelniania dla RDP, co sprawia, że PowerShell Remoting jest bezpieczniejszą alternatywą.
- `Digest` — jest jedną z metod, które serwer WWW może wykorzystać do uwierzytelniania użytkowników. Zanim nazwa użytkownika i hasło zostaną przesłane przez sieć za pomocą protokołu **HTTP**, są haszowane przy użyciu algorytmu kryptograficznego **MD5**, a przed haszowaniem dodawana jest jednorazowa wartość *nonce*, co pozwala na unikanie ataków typu replay.

  Choć `Digest` nie zapewnia tak silnego uwierzytelniania jak inne protokoły oparte na kluczach, to jest bezpieczniejszą metodą niż inne, słabsze mechanizmy uwierzytelniania i stanowi lepszą alternatywę dla słabego uwierzytelniania podstawowego.
- `Kerberos` — uwierzytelnianie w tej formie wykorzystuje protokół Kerberos, który jest standardem w środowiskach domenowych i zapewnia najwyższy poziom bezpieczeństwa.
- `Negotiate` — ta opcja pozwala klientowi na negocjowanie metody uwierzytelniania. Jeżeli używane jest konto domenowe, autoryzacja odbywa się za pomocą protokołu Kerberos; w przypadku kont lokalnych uwierzytelnianie wykorzystuje protokół NTLM.
- `NegotiateWithImplicitCredential` — ta metoda wykorzystuje do uwierzytelniania (uruchamiania jako) poświadczenia bieżącego użytkownika.

Wymienione wyżej mechanizmy uwierzytelniania mogą być używane we wszystkich cmdletach PowerShell Remoting.

Krótki opis tych metod znajdziesz również w dokumentacji firmy Microsoft na stronie *https://learn.microsoft.com/en-us/dotnet/api/system.management.automation.runspaces.authenticationmechanism*.

Pamiętaj, że PowerShell klasyfikuje niektóre mechanizmy uwierzytelniania jako potencjalnie niebezpieczne, co może skutkować wyświetlaniem ostrzeżeń i komunikatów o błędach podczas próby ich użycia. W takich sytuacjach konieczne jest ich jawne potwierdzenie i zignorowanie, tak aby umożliwić kontynuowanie korzystania z wybranego, potencjalnie niebezpiecznego mechanizmu uwierzytelniania.

Zagadnienia związane z bezpieczeństwem uwierzytelniania typu Basic

Bez dodatkowych warstw szyfrowania uwierzytelnianie typu `Basic` jest narażone na ogromne ryzyko i stwarza poważne zagrożenia bezpieczeństwa. W tej sekcji przekonasz się, że nie powinno się korzystać z takiego uwierzytelniania bez dodatkowego szyfrowania, i dowiesz się, dlaczego zawsze należy stosować szyfrowanie za pomocą protokołu **TLS** (*Transport Layer Security*) w sytuacjach, gdzie użycie uwierzytelniania typu `Basic` jest konieczne.

Uwaga

Nie używaj tej konfiguracji w środowisku produkcyjnym, ponieważ jest ona wysoce niebezpieczna i została zaprezentowana jedynie do celów testowych. Użycie tej konfiguracji w praktyce może narazić Twój system na poważne ryzyko i zagrożenie bezpieczeństwa.

Aby skonfigurować **środowisko testowe** do korzystania z uwierzytelniania typu `Basic` i przesyłania nieszyfrowanego ruchu sieciowego, powinieneś tak zmodyfikować ustawienia WinRM, aby akceptowało te oba elementy.

W tym przykładzie zdalnym hostem, z którym chcemy się połączyć, jest `PSSec-PC01`, a komunikacja odbywa się z wykorzystaniem nieszyfrowanego ruchu sieciowego i uwierzytelniania typu `Basic`. Połączenie zostanie nawiązane z maszyny zarządzającej o nazwie `PSSec-PC02`.

Próbując nawiązać połączenie z `PSSec-PC02` do `PSSec-PC01` (adres IP `172.29.0.12`) przy użyciu parametru `-Authentication Basic`, otrzymamy komunikat informujący o konieczności podania nazwy użytkownika i hasła w celu dokonania uwierzytelnienia typu `Basic`, tak jak pokazano na rysunku 3.18.

Po wprowadzeniu poświadczeń logowania nadal nie będziemy w stanie się uwierzytelnić i otrzymamy kolejny komunikat o błędzie informujący o braku dostępu. Wynika to z faktu, że uwierzytelnianie typu `Basic` jest niezabezpieczonym mechanizmem uwierzytelniania, jeśli nie jest dodatkowo chronione przez protokół TLS. Z tego powodu PowerShell Remoting nie pozwala na nawiązanie połączenia przy użyciu tego niezabezpieczonego mechanizmu, chyba że zostanie on jawnie skonfigurowany.

```
PS C:\Users\Administrator> New-PSSession -ComputerName 172.29.0.12 -Authentication Basic
New-PSSession: The WinRM client cannot process the request. Requests must include user name and password when
 Basic or Digest authentication mechanism is used. Add the user name and password or change the authenticatio
n mechanism and try the request again.
PS C:\Users\Administrator> $cred = Get-Credential -Credential "PSSec"

PowerShell credential request
Enter your credentials.
Password for user PSSec: ****************

PS C:\Users\Administrator> New-PSSession -ComputerName 172.29.0.12 -Authentication Basic -Credential $cred
New-PSSession: [172.29.0.12] Connecting to remote server 172.29.0.12 failed with the following error message
: Access is denied. For more information, see the about_Remote_Troubleshooting Help topic.
PS C:\Users\Administrator>
```

Rysunek 3.18. W przypadku użycia niezabezpieczonego mechanizmu uwierzytelniania wyświetlane są komunikaty o błędach

Spróbujmy zatem jawnie skonfigurować uwierzytelnianie typu `Basic` w naszym środowisku testowym, mając świadomość, że w ten sposób celowo obniżamy poziom zabezpieczeń naszej konfiguracji. Najpierw zezwolimy na nieszyfrowany ruch na `PSSec-PC01`.

```
> winrm set winrm/config/service '@{AllowUnencrypted="true"}'
```

Pamiętaj o rozróżnieniu między konfiguracją usługi a konfiguracją klienta. Ponieważ chcemy połączyć się z komputerem `PSSec-PC01`, dokonujemy konfiguracji usługi WinRM, a nie klienta.

W kolejnym kroku skonfigurujemy usługę tak, aby akceptowała uwierzytelnianie typu `Basic`.

```
> winrm set winrm/config/service/auth '@{Basic="true"}'
```

Po wprowadzeniu zmian w konfiguracji WinRM musisz ją ponownie uruchomić, aby nowa konfiguracja zaczęła obowiązywać:

```
> Restart-Service -Name WinRM
```

Teraz skonfigurujmy komputer `PSSec-PC02` do nawiązywania nieszyfrowanych połączeń z innymi urządzeniami przy użyciu uwierzytelniania typu `Basic`.

Najpierw musimy skonfigurować klienta, aby można było zainicjować połączenia nieszyfrowane:

```
> winrm set winrm/config/client '@{AllowUnencrypted="true"}'
```

Następnie musimy upewnić się, że klient może nawiązywać połączenia przy użyciu uwierzytelniania typu `Basic`:

```
> winrm set winrm/config/client/auth '@{Basic="true"}'
```

Na koniec musimy ponownie uruchomić usługę WinRM, aby załadować nową konfigurację:

```
> Restart-Service -Name WinRM
```

Pamiętaj, że taka konfiguracja naraża komputery na ogromne ryzyko i czyni je bardzo podatnymi na ataki, a w szczególności może ujawnić dane uwierzytelniające potencjalnym napastnikom, którzy mogą przechwytywać Twój ruch sieciowy, kiedy łączysz się z innymi komputerami. Może to pozwolić napastnikowi na uzyskanie nieautoryzowanego dostępu do atakowanych systemów i doprowadzić do poważnego incydentu bezpieczeństwa związanego z potencjalnym wyciekiem poufnych danych lub wykonaniem innych złośliwych działań.

Z tego względu taką konfigurację powinieneś stosować wyłącznie w kontrolowanym środowisku testowym. W środowiskach produkcyjnych niezbędne jest podjęcie odpowiednich środków bezpieczeństwa, takich jak implementacja szyfrowania oraz wykorzystanie bezpiecznych protokołów uwierzytelniania, w celu zapewnienia odpowiedniej ochrony urządzeń i bezpieczeństwa danych.

Po wprowadzeniu tej wrażliwej konfiguracji nadszedł czas na nawiązanie połączenia wykorzystującego uwierzytelnianie typu `Basic`. Na potrzeby tego przykładu na komputerze `PSSec-PC01` utworzyłam konto użytkownika o nazwie `PSSec`.

Spróbujemy teraz nawiązać połączenie z `PSSec-PC02` do `PSSec-PC01` (adres IP tego komputera to `172.29.0.12`) z wykorzystaniem parametru `-Authentication Basic` oraz poświadczeniami dla użytkownika `PSSec`.

```
> $cred = Get-Credential -Credential "PSSec"
> New-PSSession -ComputerName 172.29.0.12 -Authentication Basic -Credential $cred
```

Nasza sesja jest inicjowana. Jeżeli teraz za pomocą Wiresharka prześledzimy ruch sieciowy związany z nawiązywaniem tego połączenia, zobaczymy przesyłane żądania SOAP. Co gorsza, możemy również zobaczyć nagłówek `Authorization`, w którym znajdują się zakodowane za pomocą algorytmu Base64 nazwa użytkownika i hasło, tak jak pokazano na rysunku 3.19.

```
POST /wsman?PSVersion=5.1.17763.1490 HTTP/1.1
Connection: Keep-Alive
Content-Type: application/soap+xml;charset=UTF-8
User-Agent: Microsoft WinRM Client
Content-Length: 1667
Host: 172.29.0.12:5985
Authorization: Basic UFNTZWM6UFMtU2VjUm9ja3oxMjM0IQ==

<s:Envelope xmlns:s="http://www.w3.org/2003/05/soap-envelope" xmlns:a="http://schemas.xmlsoap.org/ws/2004/08/addressing"
xmlns:w="http://schemas.dmtf.org/wbem/wsman/1/wsman.xsd" xmlns:p="http://schemas.microsoft.com/wbem/wsman/1/
wsman.xsd"><s:Header><a:To>http://172.29.0.12:5985/wsman?PSVersion=5.1.17763.1490</a:To><w:ResourceURI
s:mustUnderstand="true">http://schemas.microsoft.com/powershell/Microsoft.PowerShell</w:ResourceURI><a:ReplyTo><a:Address
s:mustUnderstand="true">http://schemas.xmlsoap.org/ws/2004/08/addressing/role/anonymous</a:Address></a:ReplyTo><a:Action
s:mustUnderstand="true">http://schemas.microsoft.com/wbem/wsman/1/windows/shell/Receive</a:Action><w:MaxEnvelopeSize
s:mustUnderstand="true">512000</w:MaxEnvelopeSize><a:MessageID>uuid:364E780F-DBA3-4E81-A5E6-E702FDF0DDAE</
a:MessageID><w:Locale xml:lang="en-US" s:mustUnderstand="false" /><p:DataLocale xml:lang="de-DE" s:mustUnderstand="false" /
><p:SessionId s:mustUnderstand="false">uuid:65990572-516D-427E-A2AB-93614F276205</p:SessionId><p:OperationID
s:mustUnderstand="false">uuid:AA660375-7929-4BF7-B1D1-4D4E7AF7D946</p:OperationID><p:SequenceId
s:mustUnderstand="false">1</p:SequenceId><w:SelectorSet><w:Selector Name="ShellId">06483EB8-EF4B-43BF-815C-E3DD34CEADEB</
w:Selector></w:SelectorSet><w:OptionSet xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"><w:Option
Name="WSMAN_CMDSHELL_OPTION_KEEPALIVE">TRUE</w:Option></w:OptionSet><w:OperationTimeout>PT180.000S</w:OperationTimeout></
s:Header><s:Body><rsp:Receive xmlns:rsp="http://schemas.microsoft.com/wbem/wsman/1/windows/shell"
SequenceId="0"><rsp:DesiredStream>stdout</rsp:DesiredStream></rsp:Receive></s:Body></s:Envelope>HTTP/1.1 200
Content-Type: application/soap+xml;charset=UTF-8
Server: Microsoft-HTTPAPI/2.0
Date: Sun, 10 Jan 2021 15:29:54 GMT
Content-Length: 2264

<s:Envelope xml:lang="en-US" xmlns:s="http://www.w3.org/2003/05/soap-envelope" xmlns:a="http://schemas.xmlsoap.org/ws/
2004/08/addressing" xmlns:w="http://schemas.dmtf.org/wbem/wsman/1/wsman.xsd" xmlns:rsp="http://schemas.microsoft.com/wbem/
wsman/1/windows/shell" xmlns:p="http://schemas.microsoft.com/wbem/wsman/1/wsman.xsd"><s:Header><a:Action>http://
schemas.microsoft.com/wbem/wsman/1/windows/shell/ReceiveResponse</a:Action><a:MessageID>uuid:FFB2E649-B93F-4125-B4DE-
F0EE9C18C67E</a:MessageID><a:To>http://schemas.xmlsoap.org/ws/2004/08/addressing/role/anonymous</a:To><p:OperationID
s:mustUnderstand="false">uuid:AA660375-7929-4BF7-B1D1-4D4E7AF7D946</p:OperationID><p:SequenceId>1</
p:SequenceId><a:RelatesTo>uuid:364E780F-DBA3-4E81-A5E6-E702FDF0DDAE</a:RelatesTo></
s:Header><s:Body><rsp:ReceiveResponse><rsp:Stream
Name="stdout">3gDeAAAAAAAAAAABAAAAAAAAAAADAAAAygEAAAACAAEAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAADvu788T2JqIFJlZklkPSIwI
j48TVM+PFZlcnNpb24gTj0icHJvdG9jb2x2ZXJzaW9uIj4yLjM8L1ZlcnNpb24+PFZlcnNpb24gTj0iUFNWZXJzaW9uIj4yLjA8L1ZlcnNpb24+PFZlcnNpb24g
```

Rysunek 3.19. Przechwytywanie za pomocą Wiresharka ruchu sieciowego z uwierzytelnianiem typu Basic

Dane zakodowane przy użyciu algorytmu Base64 można łatwo odkodować, na przykład za pomocą PowerShella, tak jak pokazano na rysunku 3.20.

```
PS C:\Users\Administrator> [System.Text.Encoding]::UTF8.GetString([System.Convert]::FromBase64String
("UFNTZWM6UFMtU2VjUm9ja3oxMjM0IQ=="))
PSSec:PS-SecRockz1234!
```

Rysunek 3.20. Odkodowywanie danych uwierzytelniających zakodowanych przy użyciu algorytmu Base64

Jeżeli napastnik przechwyci taki ruch sieciowy, może łatwo odkryć, że hasło użytkownika `PSSec` to `PS-SecRockz1234!`, a następnie może przeprowadzić atak typu *man-in-the-middle* lub użyć tego hasła do podszycia się pod użytkownika `PSSec`, co jest doskonałym punktem wyjścia do kompromitacji całego środowiska.

Mam nadzieję, że udało mi się w klarowny sposób przedstawić zagrożenia związane z używaniem uwierzytelniania typu `Basic` oraz nieszyfrowanych sesji, zachęcić Cię do stosowania tej konfiguracji jedynie w środowiskach testowych i przekonać do unikania takich rozwiązań w środowiskach produkcyjnych.

Kradzież poświadczeń w kontekście PowerShell Remoting

W zależności od zastosowanej metody uwierzytelniania dane uwierzytelniające mogą być przekazywane do zdalnego systemu, który może zostać przejęty przez napastnika. Jeżeli chcesz dowiedzieć się więcej o **kradzieży danych uwierzytelniających** oraz metodach jej zapobiegania, powinieneś zajrzeć do świetnego opracowania, zatytułowanego *Mitigating Pass-the-Hash (PtH) Attacks and Other Credential Theft* — zapobieganie atakom typu Pass-the-Hash (PtH) i innym formom kradzieży poświadczeń — który możesz pobrać ze strony *https://www.microsoft.com/en-us/download/details.aspx?id=36036*.

Domyślnie, PSRemoting nie przechowuje poświadczeń w systemie docelowym, co sprawia, że PowerShell jest wyjątkowo bezpiecznym narzędziem administracyjnym. Jednak w przypadku używania PSRemoting z uwierzytelnianiem CredSSP poświadczenia są przekazywane do systemu zdalnego, gdzie mogą zostać wyodrębnione przez napastnika i wykorzystane do przejęcia tożsamości.

Pamiętaj, że jeżeli używasz uwierzytelniania `CredSSP`, poświadczenia używane do uwierzytelniania są buforowane w systemie zdalnym. Choć takie rozwiązanie ułatwia pojedyncze logowania, to jednocześnie zdecydowanie zwiększa ryzyko kradzieży buforowanych poświadczeń logowania. Z tego względu, jeżeli to możliwe, powinieneś unikać korzystania z `CredSSP` jako mechanizmu uwierzytelniania. Jeżeli jednak zdecydujesz się na jego użycie, powinieneś włączyć mechanizm Credential Guard, aby zmniejszyć to ryzyko.

Więcej szczegółowych informacji na temat uwierzytelniania typu `CredSSP` i ataków typu pass-the-hash znajdziesz w rozdziale 6. „Active Directory — ataki i przeciwdziałanie”.

Wykonywanie poleceń przy użyciu PowerShell Remoting

Załóżmy, że pojawiła się potrzeba zdalnego uruchomienia polecenia, a PowerShell Remoting nie jest jeszcze odpowiednio skonfigurowany w Twoim systemie. Na szczęście wiele poleceń cmdlet oferuje wbudowane technologie zdalnego uruchamiania, które w takiej sytuacji możesz wykorzystać.

Wszystkie polecenia cmdlet z wbudowaną technologią zdalnego uruchamiania mają wspólną cechę: zazwyczaj posiadają parametr o nazwie `-ComputerName`, za pomocą którego możesz określić zdalny punkt końcowy.

Aby uzyskać listę poleceń cmdlet, które umożliwiają zdalne uruchamianie zadań, powinieneś użyć polecenia `Get-Command -CommandType Cmdlet -ParameterName ComputerName`, tak jak pokazano w przykładzie poniżej.

```
> Get-Command -ParameterName ComputerName
CommandType  Name               Version     Source
-----------  ----               -------     ------
Cmdlet       Connect-PSSession  3.0.0.0     Microsoft.PowerShell.Core
Cmdlet       Enter-PSSession    3.0.0.0     Microsoft.PowerShell.Core
Cmdlet       Get-PSSession      3.0.0.0     Microsoft.PowerShell.Core
Cmdlet       Invoke-Command     3.0.0.0     Microsoft.PowerShell.Core
Cmdlet       New-PSSession      3.0.0.0     Microsoft.PowerShell.Core
Cmdlet       Receive-Job        3.0.0.0     Microsoft.PowerShell.Core
Cmdlet       Receive-PSSession  3.0.0.0     Microsoft.PowerShell.Core
Cmdlet       Remove-PSSession   3.0.0.0     Microsoft.PowerShell.Core
```

Pamiętaj, że przedstawiony przykład jest jedynie ilustracją, a zaprezentowana lista oczywistych powodów nie jest kompletna.

Polecenia cmdlet z parametrem `-ComputerName` nie zawsze wykorzystują WinRM. W zależności od technologii bazowej niektóre polecenia używają WMI, a inne RPC.

Każde polecenie cmdlet ma przypisany konkretny protokół, co oznacza, że konfiguracja zapory i usług musi być dostosowana do wymagań danego protokołu. Może to wprowadzać znaczny narzut na zarządzanie. W związku z tym w przypadku zdalnego zarządzania środowiskami warto w optymalny sposób skonfigurować PSRemoting — korzystanie z WinRM jest zdecydowanie bardziej przyjazne dla zapory sieciowej, a jej konfiguracja i utrzymanie są znacznie łatwiejsze.

Nie pomyl pojęć!

PowerShell Remoting i użycie parametru `-ComputerName` w poleceniach cmdlet to dwie różne metody zdalnego wykonywania poleceń, które oferują odmienne możliwości i zastosowania. Polecenia cmdlet z parametrem `-ComputerName` działają na podstawie swoich protokołów, które często wymagają utworzenia osobnych reguł zapory sieciowej, aby mogły działać poprawnie.

Wykonywanie pojedynczych poleceń i bloków skryptów

Za pomocą polecenia cmdlet `Invoke-Command` można *wykonać pojedyncze polecenie* lub *całe bloki skryptów* na komputerze zdalnym lub lokalnym. Składnia tego polecenia jest następująca:

```
Invoke-Command -ComputerName <NazwaKomputera> -ScriptBlock {<BlokSkryptu>}
```

W kolejnym przykładzie możesz zobaczyć, jak zrestartować bufor drukarki na komputerze zdalnym `PSSec-PC01` i wyświetlić pełne wyniki działania takiego polecenia (opcja `-Verbose`):

```
> Invoke-Command -ComputerName PSSec-PC01 -ScriptBlock { Restart-Service
➥-Name Spooler -Verbose }
VERBOSE: Performing the operation "Restart-Service" on target "Print Spooler (Spooler)".
```

Polecenie `Invoke-Command` to świetny sposób na uruchamianie lokalnych skryptów i poleceń na zdalnym komputerze.

Jeśli nie chcesz kopiować skryptów na zdalne maszyny, możesz użyć polecenia `Invoke-Command` z parametrem `-FilePath`, aby *uruchomić lokalny skrypt w zdalnym systemie*:

```
> Invoke-Command -ComputerName PSSec-PC01 -FilePath c:\tmp\test.ps1
```

Jeżeli korzystasz z parametru `-FilePath` w poleceniu `Invoke-Command`, wszelkie zależności wymagane przez skrypt (takie jak inne skrypty, moduły czy polecenia) są również dostępne w systemie zdalnym — w przeciwnym razie skrypt może nie działać zgodnie z oczekiwaniami bądź próba jego wykonania zakończy się niepowodzeniem.

Polecenia lub *bloki skryptów* mogą być również uruchamiane w wielu systemach jednocześnie. Aby to zrobić, wystarczy w parametrze `-ComputerName` wskazać wszystkie zdalne komputery, na których polecenie lub skrypt mają zostać wykonane. Na przykład poniższe polecenie restartuje bufor wydruku na komputerach `PSSec-PC01` i `PSSec-PC02`:

```
> Invoke-Command -ComputerName PSSec-PC01,PSSec-PC02 {Restart-Service -Name Spooler}
```

Więcej szczegółowych informacji na temat opcji i parametrów wywołania polecenia `Invoke-Command` znajdziesz w oficjalnej dokumentacji PowerShella na stronie *https://docs.microsoft.com/pl-pl/powershell/module/microsoft.powershell.core/invoke-command*.

Praca z sesjami PowerShell

Parametr `-Session` wskazuje, że cmdlet lub funkcja obsługuje sesje w ramach PowerShell Remoting.

Aby znaleźć w swoim systemie wszystkie polecenia, które obsługują parametr `-Session`, możesz użyć polecenia `Get-Command -ParameterName session`, tak jak pokazano na rysunku 3.21.

Wykonanie takiego polecenia spowoduje wyświetlenie wszystkich poleceń dostępnych w Twoim systemie, które obsługują parametr `-Session`.

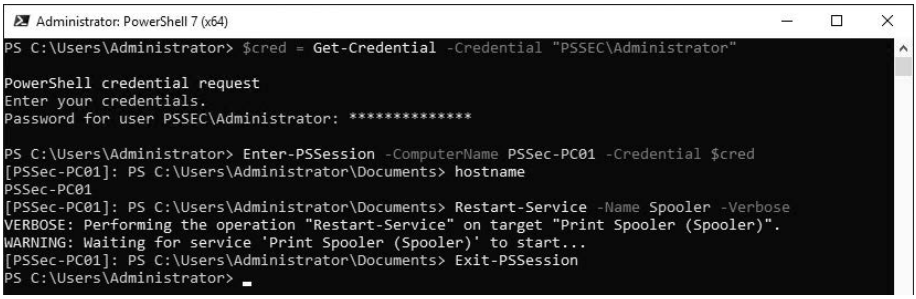
```
PS C:\Users\Administrator> Get-Command -ParameterName session

CommandType     Name                                               Version    Source
-----------     ----                                               -------    ------
Cmdlet          Connect-PSSession                                  7.2.9.500  Microsoft.PowerShell…
Cmdlet          Disconnect-PSSession                               7.2.9.500  Microsoft.PowerShell…
Cmdlet          Enter-PSSession                                    7.2.9.500  Microsoft.PowerShell…
Cmdlet          Invoke-Command                                     7.2.9.500  Microsoft.PowerShell…
Cmdlet          New-PSSession                                      7.2.9.500  Microsoft.PowerShell…
Cmdlet          Receive-Job                                        7.2.9.500  Microsoft.PowerShell…
Cmdlet          Receive-PSSession                                  7.2.9.500  Microsoft.PowerShell…
Cmdlet          Remove-PSSession                                   7.2.9.500  Microsoft.PowerShell…
```

Rysunek 3.21. Wyświetlanie poleceń, które posiadają parametr session

Sesje interaktywne

Wykorzystując polecenie `Enter-PSSession`, można zainicjować sesję interaktywną. Po ustanowieniu sesji można pracować w powłoce systemu zdalnego. Przykład takiego polecenia został pokazany na rysunku 3.22.



Rysunek 3.22. Nawiązanie zdalnej, interaktywnej sesji PowerShell, wykonanie polecenia i zakończenie sesji

Po zakończeniu pracy powinieneś użyć polecenia `Exit-PSSession`, aby zamknąć sesję i połączenie zdalne.

Sesje trwałe

Polecenie cmdlet `New-PSSession` może być wykorzystane do ustanowienia sesji trwałej (ang. *persistent session*).

Podobnie jak w poprzednim przykładzie, używamy polecenia `Get-Credential` do przechowywania poświadczeń logowania w postaci bezpiecznego ciągu zapisanego w zmiennej `$cred`.

Za pomocą polecenia pokazanego poniżej tworzymy dla komputerów zdalnych `PSSec-PC01` i `PSSec-PC02` dwie sesje, które następnie zostaną wykorzystane do wykonywania poleceń.

```
$sessions = New-PSSession -ComputerName PSSec-PC01, PSSec-PC02 -Credential $cred
```

Aby wyświetlić wszystkie aktywne sesje, możesz użyć polecenia `Get-PSSession`, tak jak pokazano na rysunku 3.23.

```
PS C:\Windows\System32> $cred = Get-Credential -Credential "PSSEC\Administrator"

PowerShell credential request
Enter your credentials.
Password for user PSSEC\Administrator: **************

PS C:\Windows\System32> $sessions = New-PSSession -ComputerName PSSec-PC01, PSSec-PC02 -Credential
 $cred
PS C:\Windows\System32> Get-PSSession

 Id Name            Transport ComputerName    ComputerType    State         ConfigurationName
 -- ----            --------- ------------    ------------    -----         -----------------
  1 Runspace1       WSMan     PSSec-PC01      RemoteMachine   Opened        Microsoft.PowerShell
  2 Runspace2       WSMan     PSSec-PC02      RemoteMachine   Opened        Microsoft.PowerShell
```

Rysunek 3.23. Tworzenie sesji trwałych i wyświetlanie listy aktywnych sesji

Teraz możesz użyć zmiennej `$sessions` do uruchamiania poleceń we wszystkich sesjach zdalnych, które zostały wcześniej utworzone.

Jednym z częstych zastosowań jest sprawdzenie, czy wszystkie bądź wybrane aktualizacje zabezpieczeń zostały zainstalowane na komputerach zdalnych. W poniższym przykładzie przedstawiono metodę sprawdzenia, czy poprawka `KB5023773` została zainstalowana na wszystkich maszynach zdalnych. Aby uniknąć wyświetlania komunikatów o błędach, jeśli poprawka nie została odnaleziona, użyty został parametr `-ErrorAction SilentlyContinue`:

```
Invoke-Command –Session $sessions -ScriptBlock { Get-Hotfix -Id 'KB5023773' -
↪ErrorAction SilentlyContinue }
```

Na rysunku 3.24 pokazano przykładowe wyniki działania takiego polecenia.

```
PS C:\Windows\System32> Invoke-Command -Session $sessions -ScriptBlock { Get-Hotfix -Id 'KB5023773' -ErrorAction
SilentlyContinue }

Source        Description      HotFixID      InstalledBy          InstalledOn                PSComputerName
------        -----------      --------      -----------          -----------                --------------
PSSEC-PC01    Update           KB5023773     NT AUTHORITY\SYSTEM  06.04.2023 00:00:00        PSSec-PC01

PS C:\Windows\System32>
```

Rysunek 3.24. Uruchamianie polecenia we wszystkich utworzonych wcześniej sesjach

Jak łatwo zauważyć, hotfix jest zainstalowany tylko na `PSSec-PC01`, ale brakuje go na drugim komputerze, `PSSec-02`.

Aby zainstalować brakującą aktualizację, możemy wybrać jedną z dwóch metod: przesłać dodatkowe polecenia bezpośrednio do sesji lub przejść do sesji w trybie interaktywnym, podając identyfikator sesji za pomocą polecenia `Enter-PSSession -Id 2`, tak jak pokazano na rysunku 3.25.

```
PS C:\Windows\System32> Enter-PSSession -Id 2
[PSSec-PC02]: PS C:\Users\Administrator\Documents> Get-WindowsUpdate -Install -KBArticleID 'KB5023773'
[PSSec-PC02]: PS C:\Users\Administrator\Documents> Exit-PSSession
PS C:\Windows\System32>
```

Rysunek 3.25. Wejście do trwałej sesji, uruchomienie polecenia i wyjście z sesji

Po wejściu do sesji możemy uruchomić polecenie `Get-WindowsUpdate`, aby zainstalować brakującą aktualizację. Pamiętaj, że polecenie to nie jest domyślnie dostępne i wymaga zainstalowania modułu `PSWindowsUpdate`:

```
Get-WindowsUpdate -Install -KBArticleID 'KB5023773'
```

Po zainstalowaniu aktualizacji możemy zakończyć sesję, używając polecenia `Exit-PSSession`. Pamiętaj, że spowoduje to tylko odłączenie od sesji, ale pozostawi ją otwartą.

Uwaga

Jeżeli używasz sesji interaktywnej, wszystkie wykonywane moduły, takie jak `PSWindows Update`, muszą być zainstalowane w zdalnym systemie. Jeżeli jednak używasz polecenia `Invoke-Command` do uruchamiania poleceń w sesji trwałej, to wystarczy, że moduł będzie zainstalowany tylko na komputerze, z którego uruchamiasz polecenia.

```
Invoke-Command – Session $sessions -ScriptBlock { Get-WindowsUpdate -
↪Install -KBArticleID 'KB5023773'}
```

Jeżeli teraz ponownie sprawdzimy aktualizację `KB5023773`, zobaczymy, że jest już zainstalowana na obu komputerach testowych, tak jak pokazano na rysunku 3.26.

```
PS C:\Windows\System32> Invoke-Command -Session $sessions -ScriptBlock { Get-Hotfix -Id 'KB5023773' -ErrorAction
 SilentlyContinue }

Source        Description      HotFixID      InstalledBy          InstalledOn                PSComputerName
------        -----------      --------      -----------          -----------                --------------
PSSEC-PC01    Update           KB5023773     NT AUTHORITY\SYSTEM  06.04.2023 00:00:00        PSSec-PC01
PSSEC-PC02    Update           KB5023773     NT AUTHORITY\SYSTEM  06.04.2023 00:00:00        PSSec-PC02

PS C:\Windows\System32>
```

Rysunek 3.26. Aktualizacja została zainstalowana pomyślnie

Kiedy zakończymy pracę i nie potrzebujemy już zdalnych sesji, możemy je usunąć za pomocą polecenia `Remove-PSSession`:

- Aby usunąć wszystkie sesje, które utworzyliśmy w naszym przykładzie, możemy użyć zmiennej `$sessions`:

  ```
  Remove-PSSession -Session $sessions
  ```

- Zamiast tego możemy usunąć pojedynczą, wybraną sesję, wskazując ją za pomocą parametru `-id`:

  ```
  Remove-PSSession -id 2
  ```

Po usunięciu jednej lub wszystkich sesji możesz użyć polecenia `Get-PSSession`, aby to zweryfikować, tak jak pokazano na rysunku 3.27.

```
PS C:\Windows\System32> Remove-PSSession -Session $sessions
PS C:\Windows\System32> Get-PSSession
PS C:\Windows\System32>
```

Rysunek 3.27. Usuwanie wszystkich trwałych sesji

PowerShell Remoting to świetne rozwiązanie, które może znacząco zautomatyzować i uprościć Twoje codzienne zadania administracyjne. Teraz gdy opanowałeś już podstawy PSRemoting, możesz połączyć je z umiejętnością tworzenia skryptów PowerShell, by jeszcze efektywniej zarządzać systemami. Jakie wyzwania będziesz w stanie przezwyciężyć i jakie procesy zautomatyzować dzięki tej potężnej kombinacji narzędzi?

Najlepsze praktyki

Aby zagwarantować najwyższy poziom bezpieczeństwa i wydajności podczas korzystania z PSRemoting, powinieneś korzystać z najlepszych praktyk rekomendowanych przez użytkowników i deweloperów. Dzięki temu zminimalizujesz ryzyko naruszenia bezpieczeństwa i zapewnisz płynne działanie zadań związanych ze zdalnym zarządzaniem.

Uwierzytelnianie:

- Jeżeli to możliwe, korzystaj wyłącznie z uwierzytelniania Kerberos lub NTLM.
- Unikaj uwierzytelniania `CredSSP` oraz uwierzytelniania typu `Basic`, kiedy tylko to możliwe.
- W idealnym scenariuszu ogranicz stosowanie wszystkich innych mechanizmów uwierzytelniania, poza Kerberos i NTLM.
- W przypadku zdalnego zarządzania za pomocą SSH skonfiguruj uwierzytelnianie oparte na kluczu publicznym oraz zapewnij odpowiednią ochronę klucza prywatnego.

Ograniczanie połączeń:

- Za pomocą zapory sieciowej ogranicz połączenia z podsieci zarządzania (sprzętowo i programowo, jeżeli to możliwe).

  Domyślne zasady zapory sieciowej dla PSRemoting różnią się w zależności od profilu sieciowego. W *profilu domeny*, *grupy roboczej* lub *sieci prywatnej* PSRemoting jest domyślnie dostępny dla wszystkich (pod warunkiem oczywiście, że mają ważne poświadczenia logowania). W *profilu publicznym* nasłuchiwanie ruchu PSRemoting jest domyślnie blokowane. Jeżeli zostanie to wymuszone, reguła sieciowa ograniczy dostęp tylko do systemów zlokalizowanych w tej samej podsieci.
- Korzystaj z bezpiecznego systemu zarządzania do administrowania systemami za pomocą PowerShell Remoting. Jeżeli masz taką możliwość, rozważ ograniczenie dostępu z **wirtualnej sieci** zarządzającej (**VNet**), co pomoże zwiększyć bezpieczeństwo, a także zastosuj to podejście do innych protokołów zarządzania, takich jak RDP, WMI, CIM i podobne.
- Przy tworzeniu systemu zarządzania wykorzystaj zasadę czystego źródła i zastosuj zalecany model dostępu uprzywilejowanego:
  - *https://learn.microsoft.com/pl-pl/security/privileged-access-workstations/privileged-access-success-criteria#clean-source-principle*
  - *https://learn.microsoft.com/pl-pl/security/privileged-access-workstations/privileged-access-access-model*

Ograniczanie sesji:

- Używaj trybu ograniczonego języka (ang. *Constrained Language Mode*) i JEA.
- Więcej informacji na temat JEA, trybu ograniczonego języka, bezpieczeństwa sesji i SDDL (ang. *Security Descriptor Definition Language*) znajdziesz w rozdziale 10. „Tryby językowe sesji PowerShell i funkcjonalność JEA”.

Sprawdzanie poprawności ustawień:

- Użyj zasad grupy dla WinRM, aby wymusić bezpieczne ustawienia PowerShell Remoting we wszystkich zarządzanych systemach. Uwzględnij wymagania dotyczące szyfrowania danych oraz uwierzytelniania, aby zapewnić najwyższy poziom bezpieczeństwa komunikacji zdalnej.
- `Get-Item WSMan:\localhost\Client\AllowUnencrypted` — ta opcja nie powinna mieć wartości `$true`.
- Regularnie przeprowadzaj audyt ustawień WinRM w celu wykrycia niezabezpieczonych konfiguracji i zapewnienia zgodności z najlepszymi praktykami i zasadami bezpieczeństwa:

```
Get-Item WSMan:\localhost\client\AllowUnencrypted
Get-Item wsman:\localhost\service\AllowUnencrypted
Get-Item wsman:\localhost\client\auth\Basic
Get-Item wsman:\localhost\service\auth\Basic
```

- Używaj mechanizmu **DSC** (ang. *Desired State Configuration* — konfiguracja stanów docelowych), aby przeprowadzać audyt oraz zastosować odpowiednie ustawienia w zarządzanych systemach.

Wykorzystuj wszystkie inne metody zabezpieczeń, o których mówiliśmy lub będziemy mówić w innych rozdziałach, a w szczególności:

- Włącz rejestrowanie i transkrypcję sesji oraz monitorowanie dzienników zdarzeń. Więcej informacji na ten temat znajdziesz w rozdziale 4. „Wykrywanie — audyt i monitorowanie".
- Usuń zbędne konta administratorów lokalnych i domenowych.
- Włącz i wymuszaj podpisywanie skryptów. Więcej informacji na temat podpisywania skryptów znajdziesz w rozdziale 11. „AppLocker, kontrolowanie aplikacji i podpisywanie kodu".
- Skonfiguruj mechanizm **DSC** (ang. *Desired State Configuration*) do utwardzenia zabezpieczeń systemów i lepszego zarządzania konfiguracją.

PowerShell Remoting jest efektywnym narzędziem do zarządzania systemami. Pamiętaj jednak, że jego poziom bezpieczeństwa zależy od konfiguracji. Przy odpowiednich ustawieniach PSRemoting może być znacznie bezpieczniejszym rozwiązaniem niż logowanie interaktywne w zdalnym systemie.

Podsumowanie

Po przeczytaniu tego rozdziału powinieneś być dobrze zaznajomiony z metodami zdalnego korzystania z PowerShella za pomocą PSRemoting. Poznałeś opcje umożliwiające nawiązywanie zdalnych połączeń, co pozwala zarządzać nie tylko maszynami z systemem Windows, ale także komputerami działającymi pod kontrolą innych systemów operacyjnych, takich jak macOS i Linux.

Dowiedziałeś się również, czym są punkty końcowe, i zdobyłeś podstawową wiedzę na temat tworzenia niestandardowych punktów końcowych. Umiejętności te rozwiniesz w rozdziale 10. „Tryby językowe sesji PowerShell i funkcjonalność JEA", ale podstawy już znasz.

Następnie zdobyłeś rozległą wiedzę na temat protokołów uwierzytelniania oraz szczegółowych aspektów bezpieczeństwa związanych z ich używaniem. Wiesz już, jak łatwo nieautoryzowani użytkownicy mogą uzyskać dostęp do odszyfrowanych poświadczeń, jeżeli używany jest słaby protokół uwierzytelniania.

Teraz powinieneś być już w stanie samodzielnie skonfigurować PowerShell Remoting zarówno ręcznie, jak i za pomocą zasad grupy, co pomoże Ci odpowiednio zmodyfikować i zabezpieczyć domyślną konfigurację PSRemoting w środowisku produkcyjnym.

Nauczyłeś się wykonywać polecenia za pomocą PSRemoting, co nie tylko umożliwia uruchamianie pojedynczych poleceń na zdalnym urządzeniu, lecz także pozwala na pełną automatyzację rutynowych zadań administracyjnych.

Podczas pracy z PowerShellem, zarówno w trybie zdalnym, jak i lokalnym, kluczowe znaczenie mają audyt i monitorowanie. Wykorzystanie transkrypcji i rejestrowania zdarzeń pozwala zespołom Blue Team na skuteczne wykrywanie zagrożeń i ochronę środowiska.

Teraz gdy masz już solidną wiedzę na temat PSRemoting, w kolejnym rozdziale skupimy się na wykrywaniu i rejestrowaniu działań w PowerShellu.

Gdzie warto zajrzeć?

Aby lepiej zrozumieć wybrane zagadnienia omówione w tym rozdziale, powinieneś zapoznać się również z następującymi materiałami:

Uwierzytelnianie:

- Dokument *RFC 2617 — HTTP authentication (basic and digest authentication)*: *https://tools.ietf.org/html/rfc2617*.
- Microsoft Learn — dokumentacja *Credential Security Support Provider (CredSSP) protocol*:
  - *https://docs.microsoft.com/en-us/openspecs/windows_protocols/ms-cssp/85f57821-40bb-46aa-bfcb-ba9590b8fc30*,
  - *https://ldapwiki.com/wiki/Wiki.jsp?page=CredSSP*.
- Kryptografia klucza publicznego:
  - *https://pl.wikipedia.org/wiki/Kryptografia_klucza_publicznego*,
  - *https://www.ssh.com/ssh/public-key-authentication*.

CIM:

- *CIM cmdlet*: *https://devblogs.microsoft.com/powershell/introduction-to-cim-cmdlets/*,
- *CIM standard by DMTF*: *https://www.dmtf.org/standards/cim*.

DCOM:

- *Distributed Component Object Model (DCOM) Remote Protocol*: *https://docs.microsoft.com/en-us/openspecs/windows_protocols/ms-dcom/4a893f3d-bd29-48cd-9f43-d9777a4415b0*.

OMI:

- *Open Management Infrastructure (OMI)*: *https://cloudblogs.microsoft.com/windowsserver/2012/06/28/open-management-infrastructure/*.

Inne ciekawe źródła:

- *New-NetFirewallRule*: *https://learn.microsoft.com/en-us/powershell/module/netsecurity/new-netfirewallrule*.

PowerShell Remoting:

- *[MS-PSRP]: PowerShell Remoting protocol*: *https://learn.microsoft.com/en-us/openspecs/windows_protocols/ms-psrp/602ee78e-9a19-45ad-90fa-bb132b7cecec*.
- *Uruchamianie poleceń zdalnych*: *https://docs.microsoft.com/pl-pl/powershell/scripting/learn/remoting/running-remote-commands*.
- *Używanie komunikacji zdalnej WS-Management (WSMan) w programie PowerShell*: *https://learn.microsoft.com/pl-pl/powershell/scripting/learn/remoting/wsman-remoting-in-powershell-core*.
- *WS-Man specifications by DMTF*: *https://www.dmtf.org/standards/ws-man*.
- *Zagadnienia dotyczące zabezpieczeń komunikacji zdalnej programu PowerShell przy użyciu usługi WinRM*: *https://docs.microsoft.com/pl-pl/powershell/scripting/learn/remoting/winrmsecurity*.
- *Introduction to PowerShell Endpoints*: *https://devblogs.microsoft.com/scripting/introduction-to-powershell-endpoints/*.
- *Obsługa zdalna programu PowerShell za pośrednictwem protokołu SSH*: *https://docs.microsoft.com/pl-pl/powershell/scripting/learn/remoting/ssh-remoting-in-powershell-core*.
- *Wykonywanie drugiego przeskoku w komunikacji zdalnej programu PowerShell*: *https://docs.microsoft.com/pl-pl/powershell/scripting/learn/remoting/ps-remoting-second-hop*.

WMI:

- *Get-WmiObject*: *https://docs.microsoft.com/pl-pl/powershell/module/microsoft.powershell.management/get-wmiobject*.
- *Invoke-WmiMethod*: *https://docs.microsoft.com/pl-pl/powershell/module/microsoft.powershell.management/invoke-wmimethod*.
- *Register-WmiEvent*: *https://docs.microsoft.com/pl-pl/powershell/module/microsoft.powershell.management/register-wmievent*.
- *Remove-WmiObject*: *https://docs.microsoft.com/pl-pl/powershell/module/microsoft.powershell.management/remove-wmiobject*.
- *Set-WmiInstance*: *https://docs.microsoft.com/pl-pl/powershell/module/microsoft.powershell.management/set-wmiinstance*.

WS-Man:

- *WS-Man standard by DMTF*: *https://www.dmtf.org/standards/ws-man*.
- *Używanie komunikacji zdalnej WS-Management (WSMan) w programie PowerShell*: *https://docs.microsoft.com/pl-pl/powershell/scripting/learn/remoting/wsman-remoting-in-powershell-core*.

Wszystkie linki wymienione w tym rozdziale są dostępne w repozytorium GitHub: *https://github.com/PacktPublishing/PowerShell-Automation-and-Scripting-for-Cybersecurity/blob/master/Chapter03/Links.md*, dzięki czemu nie musisz ich wpisywać ręcznie.

Skorowidz

B

C

D

E

F

G

H

I

J

K

L

Ł

M

N

O

P

R

S

T

U

V

W

X

Z

Napastnik nie poczeka, aż opanujesz PowerShell!

PowerShell jest domyślnie instalowany w każdym nowoczesnym Windowsie. To ogromne udogodnienie dla administratorów i… potężne narzędzie w rękach cyberprzestępców. Funkcje oferowane przez PowerShell mogą zarówno zwiększyć bezpieczeństwo infrastruktury IT, jak i wspierać działania ofensywne. A zatem musisz dogłębnie poznać PowerShell!

Dzięki tej książce przyswoisz podstawy PowerShella i zasady pisania skryptów, a następnie przejdziesz do zagadnień związanych z PowerShell Remoting. Nauczysz się konfigurować i analizować dzienniki zdarzeń Windows, dowiesz się również, które zdarzenia są kluczowe do monitorowania bezpieczeństwa. Zgłębisz możliwości interakcji PowerShella z systemem operacyjnym, Active Directory i Azure AD / Entra ID. Poznasz protokoły uwierzytelniania, procesy enumeracji, metody kradzieży poświadczeń i eksploatacji, a także zapoznasz się z praktycznymi wskazówkami dla zespołów czerwonego i niebieskiego (ang. Red Team i Blue Team). Zrozumiesz też takie metody ochrony jak Just Enough Administration (JEA), AMSI, kontrola aplikacji i podpisywanie kodu.

W książce między innymi:

- użycie PowerShella do ochrony systemu i wykrywania ataków
- wgląd w dzienniki zdarzeń z poziomu PowerShella
- PSRemoting i ryzyko — obejścia i najlepsze praktyki
- uzyskiwanie dostępu do systemu, jego eksploracja i przejmowanie kontroli
- zastosowanie PowerShella w zespołach czerwonym i niebieskim
- zastosowanie JEA do ograniczania wykonywania wybranych poleceń

Miriam C. Wiesner jest starszą badaczką bezpieczeństwa w Microsofcie. Wcześniej pracowała jako administratorka i inżynier systemowy, a także jako programistka, konsultantka do spraw bezpieczeństwa i pentesterka. Jest również znaną autorką popularnych narzędzi open source opartych na PowerShellu, takich jak EventList i JEAnalyzer. Mieszka w okolicach Norymbergi w Niemczech.

Helion
helion.pl
HELION S.A.
ul. Kościuszki 1c
44-100 Gliwice
tel.: 32 230 98 63
helion@helion.pl
KOD KORZYŚCI
Sięgnij po więcej!
ISBN 978-83-289-2225-9
9 788328 922259
Cena: 119,00 zł