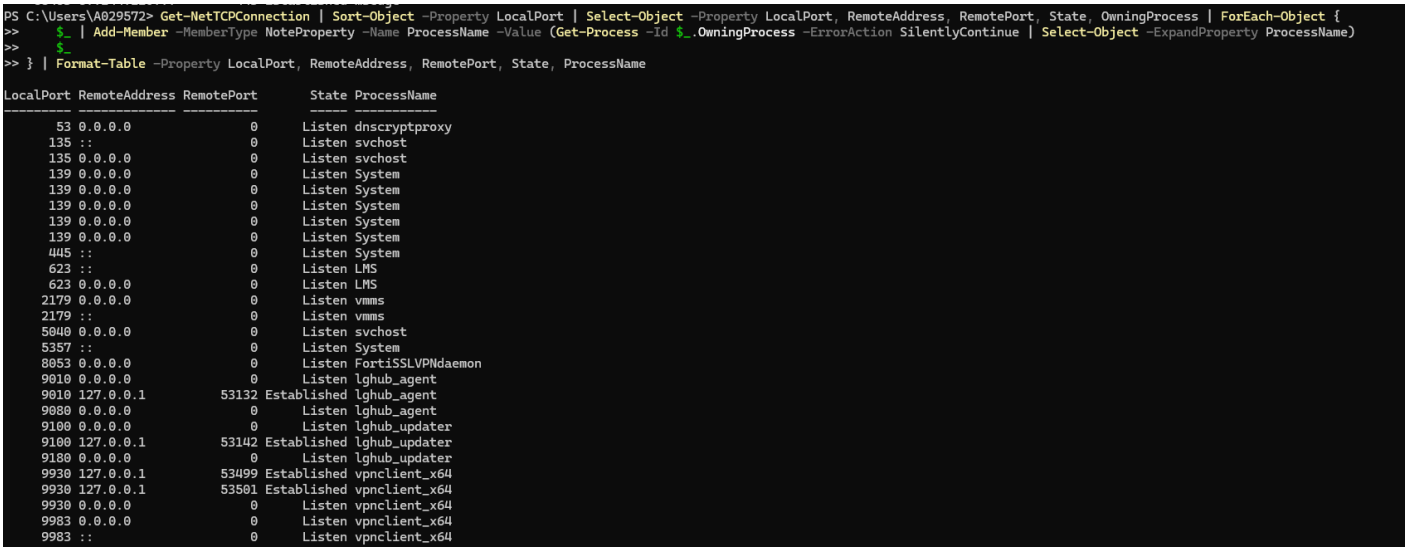
Sprawdzenie otwartych / nasłuchujących portów wraz z aktywnym procesem

Skrypty uruchamiamy w powershell.

Samo pobranie danych bez parsowania wyników:

```
Get-NetTCPConnection | Sort-Object -Property LocalPort | Select-Object -Property LocalPort, RemoteAddress, RemotePort, State, OwningProcess | ForEach-Object {
    $_ | Add-Member -MemberType NoteProperty -Name ProcessName -Value (Get-Process -Id $_.OwningProcess -ErrorAction SilentlyContinue | Select-Object -ExpandProperty ProcessName)
    $_
} | Format-Table -Property LocalPort, RemoteAddress, RemotePort, State, ProcessName
```

Przykładowy wynik:



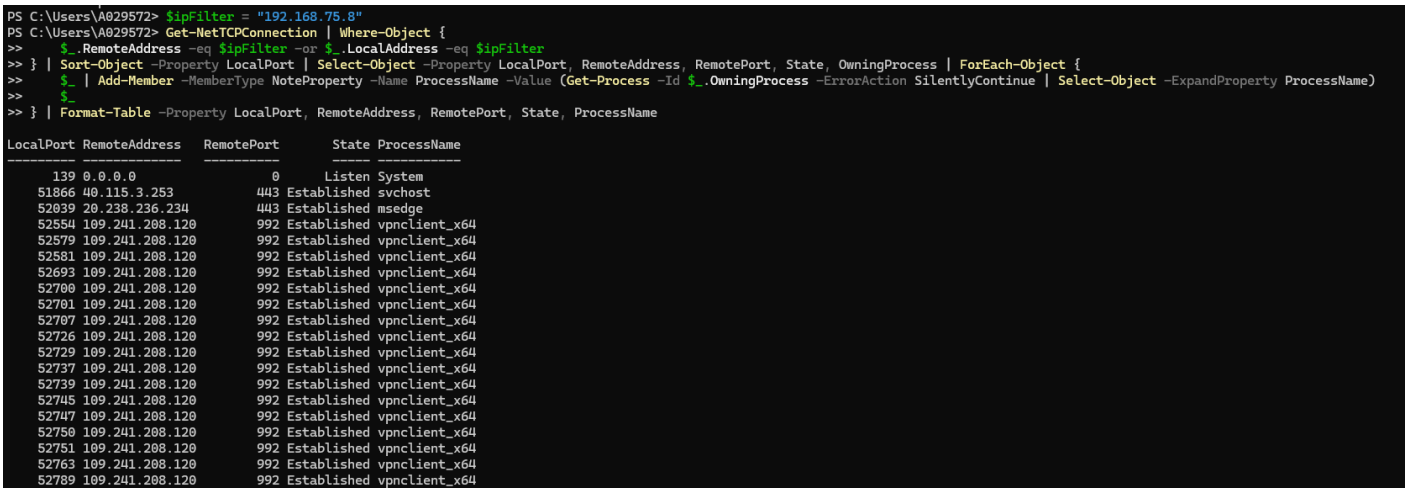
```
PS C:\Users\A029572> Get-NetTCPConnection | Sort-Object -Property LocalPort | Select-Object -Property LocalPort, RemoteAddress, RemotePort, State, OwningProcess | ForEach-Object {
>>     $_ | Add-Member -MemberType NoteProperty -Name ProcessName -Value (Get-Process -Id $_.OwningProcess -ErrorAction SilentlyContinue | Select-Object -ExpandProperty ProcessName)
>>     $_
>> } | Format-Table -Property LocalPort, RemoteAddress, RemotePort, State, ProcessName

LocalPort RemoteAddress RemotePort       State ProcessName
--------- ------------- ----------       ----- -----------
       53 0.0.0.0               0      Listen dnscryptproxy
      135 ::                    0      Listen svchost
      135 0.0.0.0               0      Listen svchost
      139 0.0.0.0               0      Listen System
      139 0.0.0.0               0      Listen System
      139 0.0.0.0               0      Listen System
      139 0.0.0.0               0      Listen System
      139 0.0.0.0               0      Listen System
      139 0.0.0.0               0      Listen System
      445 ::                    0      Listen System
      623 ::                    0      Listen LMS
      623 0.0.0.0               0      Listen LMS
     2179 0.0.0.0               0      Listen vmms
     2179 ::                    0      Listen vmms
     5040 0.0.0.0               0      Listen svchost
     5357 ::                    0      Listen System
     8053 0.0.0.0               0      Listen FortiSSLVPNdaemon
     9010 0.0.0.0               0      Listen lghub_agent
     9010 127.0.0.1         53132 Established lghub_agent
     9080 0.0.0.0               0      Listen lghub_agent
     9100 0.0.0.0               0      Listen lghub_updater
     9100 127.0.0.1         53142 Established lghub_updater
     9180 0.0.0.0               0      Listen lghub_updater
     9930 127.0.0.1         53499 Established vpnclient_x64
     9930 127.0.0.1         53501 Established vpnclient_x64
     9930 0.0.0.0               0      Listen vpnclient_x64
     9983 0.0.0.0               0      Listen vpnclient_x64
     9983 ::                    0      Listen vpnclient_x64
```

Pobieranie danych z filtrowaniem po IP:

```
$ipFilter = "Szukanie danego IP"


Get-NetTCPConnection | Where-Object {
```

```
    $_.RemoteAddress -eq $ipFilter -or $_.LocalAddress -eq $ipFilter
} | Sort-Object -Property LocalPort | Select-Object -Property LocalPort, RemoteAddress, RemotePort, State, OwningProcess | ForEach-Object {
    $_ | Add-Member -MemberType NoteProperty -Name ProcessName -Value (Get-Process -Id $_.OwningProcess -ErrorAction SilentlyContinue | Select-Object -ExpandProperty ProcessName)
    $_
} | Format-Table -Property LocalPort, RemoteAddress, RemotePort, State, ProcessName
```

Przykładowy wynik:



Pobieranie danych i zapisanie wyników do pliku

```
# Definiowanie ścieżki do pliku CSV
$csvPath = "C:\LokalizacjaDoPliku\NazwaPliku.csv"

# Pobranie wszystkich połączeń TCP i dodanie informacji o procesie
Get-NetTCPConnection | Sort-Object -Property LocalPort | Select-Object -Property LocalPort, RemoteAddress, RemotePort, State, OwningProcess | ForEach-Object {
    $_ | Add-Member -MemberType NoteProperty -Name ProcessName -Value (Get-Process -Id $_.OwningProcess -ErrorAction SilentlyContinue | Select-Object -ExpandProperty ProcessName)
    $_
} | Export-Csv -Path $csvPath -NoTypeInformation

Write-Host "Zapisano połączenia do pliku CSV: $csvPath"
```

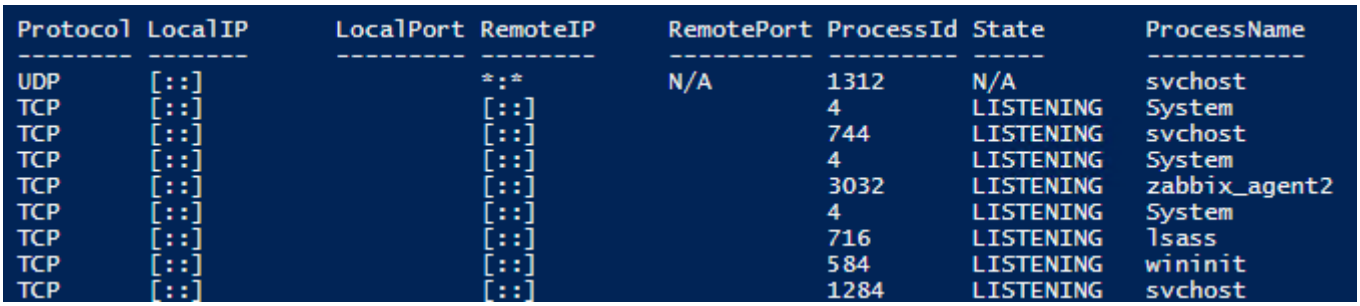
Pobieranie danych za pomocą netstat -ano z parsowaniem wyników w CLI:

```
$netstatOutput = netstat -ano
```

```
$connections = @()
foreach ($line in $netstatOutput) {
    if ($line -match '^\s*(TCP|UDP)') {
        $parts = $line -split '\s+'

        # Przypisanie wartości do odpowiednich zmiennych
        $protocol = $parts[1]
        $localAddress = $parts[2]
        $remoteAddress = $parts[3]
        $processID = ""
        $state = ""

        if ($protocol -eq 'TCP') {
            $state = $parts[4]
            $processID = $parts[5]
        } elseif ($protocol -eq 'UDP') {
            $state = "N/A"
            $processID = $parts[4]
        }

        # Funkcja do dzielenia adresu i portu, zachowująca nawiasy IPv6
        function Split-AddressPort ($address) {
            if ($address -match '^\[(.+)\]:(\d+)$') {
                # IPv6 z nawiasami np. [::1]:443
                return @("[{0}]" -f $matches[1], $matches[2])
            } elseif ($address -match '^(.+):(\d+)$') {
                # IPv4 i inne adresy z portem np. 192.168.1.1:80
                return @($matches[1], $matches[2])
            } else {
                # Adresy bez portów np. *:*
                return @($address, "N/A")
            }
        }

        $localIP, $localPort = Split-AddressPort $localAddress
        $remoteIP, $remotePort = Split-AddressPort $remoteAddress

        # Pobranie nazwy procesu lub ustawienie komunikatu
        $processName = "Unknown"
        if ($processID -eq "0") {
```

```
            $processName = "Process terminated | PID = 0"
        } elseif ($processID -match '^\d+$') {
            try {
                $processName = (Get-Process -Id $processID -ErrorAction SilentlyContinue).Name
            } catch {
                # Proces może już nie istnieć lub być niedostępny
            }
        }

        $connections += [PSCustomObject]@{
            Protocol    = $protocol
            LocalIP     = $localIP
            LocalPort   = $localPort
            RemoteIP    = $remoteIP
            RemotePort  = $remotePort
            ProcessId   = $processID
            State       = $state
            ProcessName = $processName
        }
    }
}

$connections | Sort-Object -Property LocalIP | Format-Table -AutoSize -Property Protocol, LocalIP, LocalPort,
RemoteIP, RemotePort, ProcessId, State, ProcessName
```

Przykładowy wynik:



Protocol	LocalIP	LocalPort	RemoteIP	RemotePort	ProcessId	State	ProcessName
UDP	[::]		*:*	N/A	1312	N/A	svchost
TCP	[::]		[::]		4	LISTENING	System
TCP	[::]		[::]		744	LISTENING	svchost
TCP	[::]		[::]		4	LISTENING	System
TCP	[::]		[::]		3032	LISTENING	zabbix_agent2
TCP	[::]		[::]		4	LISTENING	System
TCP	[::]		[::]		716	LISTENING	lsass
TCP	[::]		[::]		584	LISTENING	wininit
TCP	[::]		[::]		1284	LISTENING	svchost

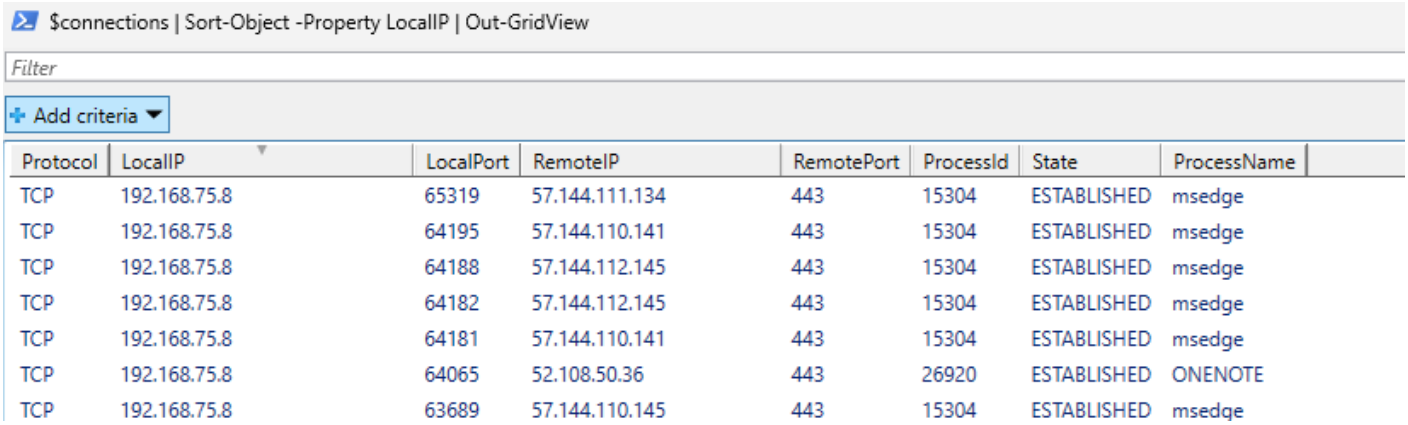
Pobieranie danych za pomocą netstat -ano z interaktywną tabelą

```
$netstatOutput = netstat -ano

$connections = @()
foreach ($line in $netstatOutput) {
    if ($line -match '^\s*(TCP|UDP)') {
```

```
$parts = $line -split '\s+'

# Przypisanie wartości do odpowiednich zmiennych
$protocol = $parts[1]
$localAddress = $parts[2]
$remoteAddress = $parts[3]
$processID = ""
$state = ""

if ($protocol -eq 'TCP') {
    $state = $parts[4]
    $processID = $parts[5]
} elseif ($protocol -eq 'UDP') {
    $state = "N/A"
    $processID = $parts[4]
}

# Funkcja do dzielenia adresu i portu, zachowująca nawiasy IPv6
function Split-AddressPort ($address) {
    if ($address -match '^\[(.+)\]:(\d+)$') {
        # IPv6 z nawiasami np. [::1]:443
        return @("[{0}]" -f $matches[1], $matches[2])
    } elseif ($address -match '^(.+):(\d+)$') {
        # IPv4 i inne adresy z portem np. 192.168.1.1:80
        return @($matches[1], $matches[2])
    } else {
        # Adresy bez portów np. *:*
        return @($address, "N/A")
    }
}

$localIP, $localPort = Split-AddressPort $localAddress
$remoteIP, $remotePort = Split-AddressPort $remoteAddress

# Pobranie nazwy procesu lub ustawienie komunikatu
$processName = "Unknown"
if ($processID -eq "0") {
    $processName = "Process terminated | PID = 0"
} elseif ($processID -match '^\d+$') {
    try {
```

```
            $processName = (Get-Process -Id $processID -ErrorAction SilentlyContinue).Name
        } catch {
            # Proces może już nie istnieć lub być niedostępny
        }
    }

    $connections += [PSCustomObject]@{
        Protocol    = $protocol
        LocalIP     = $localIP
        LocalPort   = $localPort
        RemoteIP    = $remoteIP
        RemotePort  = $remotePort
        ProcessId   = $processID
        State       = $state
        ProcessName = $processName
    }
  }
}

$connections | Sort-Object -Property LocalIP | Out-GridView
```

Przykładowy wynik:



Protocol	LocalIP	LocalPort	RemoteIP	RemotePort	ProcessId	State	ProcessName
TCP	192.168.75.8	65319	57.144.111.134	443	15304	ESTABLISHED	msedge
TCP	192.168.75.8	64195	57.144.110.141	443	15304	ESTABLISHED	msedge
TCP	192.168.75.8	64188	57.144.112.145	443	15304	ESTABLISHED	msedge
TCP	192.168.75.8	64182	57.144.112.145	443	15304	ESTABLISHED	msedge
TCP	192.168.75.8	64181	57.144.110.141	443	15304	ESTABLISHED	msedge
TCP	192.168.75.8	64065	52.108.50.36	443	26920	ESTABLISHED	ONENOTE
TCP	192.168.75.8	63689	57.144.110.145	443	15304	ESTABLISHED	msedge

Revision #1
Created 10 March 2025 14:45:38 by Tomasz Konieczny
Updated 10 March 2025 15:35:55 by Tomasz Konieczny