APPLIED INCIDENT RESPONSE

PowerShell Quick Reference

Version 20191223

Common cmdlets useful for remote triage

Cmdlet	Description
`Get-ADComputer`	Query Active Directory for computer account information. This cmdlet is found on domain controllers or can be manually added to a workstation.
`Get-ADUser`	Query information about domain user accounts. This cmdlet is auto-downloaded on domain controllers; it can be manually added to a workstation.
`Get-ChildItem`	List the items in a location, like a directory or a registry key.
`Get-CIMInstance`	Access CIM instances from a CIM server. This is the preferred way to access WMI/MI information.
`Get-Content`	Retrieve the actual contents of an object, like a file.
`Get-EventLog`	The older PowerShell way to access event logs. `Get-WinEvent` should be used instead.
`Get-HotFix`	Retrieve information about updates.
`Get-ItemProperty`	Retrieve properties, including the values of registry keys.
`Get-LocalUser`	Get information about local user accounts.
`Get-NetTCPConnection`	Query network connection information for TCP.
`Get-NetUDPEndpoint`	Query network connection information for UDP.
`Get-Process`	List information about running processes.
`Get-Service`	List information about services.
`Get-WinEvent`	Retrieve information from event logs.
`Get-WMIObject`	The older PowerShell way to access WMI objects. `Get-CIMInstance` is usually preferable.
`ForEach-Object`	Iterate over a loop for each item.
`Start-Transcript`	Record a transcript of this session to a text file, which is a great way to keep a record of your actions.
`Stop-Transcript`	Stop a previously started transcription.

Common cmdlets useful for filtering and formatting results

Cmdlet	Description
`Where-Object`	Filters the objects returned to only those that match the condition specified. Example: `Get-Process | Where-Object name -eq svchost`
`Select-Object`	Modifies the objects returned by selecting only some of their properties. Use to reduce the type of information displayed about each object. Example: `Get-Process | Select-Object ProcessName, Id`
`Sort-Object`	Sorts objects returned in ascending (default) or descending (if specified) order based on the property(ies) specified. Example: `Get-ChildItem | Sort-Object Length -Descending`
`Group-Object`	Displays object in groups based on the value of the specified property(ies). The number of items in each group are shown in the Count field.Example: `Get-Process | Group-Object Name`
`Measure-Object`	Provides counts related to the number of objects provided. Calculates the minimum, maximum, sum and average of numeric values. Calculates the number of lines, words and characters in text results. Example: `Get-NetTCPConnection | Where-Object -Property state -eq Listen | Measure-Object`
`Format-Table`	Displays the specified properties as a table. Example: `Get-ChildItem | Format-Table -Property name, LastAccessTime, LastWriteTime, CreationTime`
`Format-List`	Displays the specified properties as a list, avoiding truncating data. Example: `Get-Process | Format-List -Property name, path`
`Export-Csv`	Exports data to a file (location specified with `-Path` parameter) in a comma separated value format (the delimiter can optionally be changed). Example: `Get-Process | Export-Csv -Path "processes.csv"`
`Out-File`	Save the output to a file, specified by the `-FilePath` parameter). Example: `Get-Process | Out-File E:\capture\process.txt`
`Out-GridView`	Opens a new window where the output is displayed in a spreadsheet-like view. The column widths can be adjusted. Columns can be hidden and reordered. Data can be sorted based on specified column. Built in filtering support is also provided. Example: `Get-Process | Out-GridView`

APPLIED INCIDENT RESPONSE

PowerShell Quick Reference

Version 20191223

PowerShell Remoting

For one-to-one remoting, use the Enter-PSSession cmdlet:

```
Enter-PSSession -ComputerName Server1 -Credential admin@company.demo
```

For one-to-many remoting use Invoke-Command cmdlet:

```
Invoke-Command -ComputerName server1, server2, server3 -ScriptBlock {Get-Process |
Where-Object -Property name -eq vmtoolsd}
```

The `-ComputerName` parameter expects either a single computer name or an array of computer names. In the example above, we simply provide each computer name in a comma comma-separated list. More complex methods can be used, including providing the output of another cmdlet such as `Get-ADComputer` or using a variable to which an array of computer names has already been stored.

The `-ScriptBlock` parameter expects a PowerShell command enclosed within braces. This command will be executed on each remote system specified in the `-ComputerName` parameter. By default, the results are returned to the system where `Invoke-Command` was run. A field called PSComputerName is added by `Invoke-Command` to illustrate which of the remote systems provided each line of the response.

PowerShell for Event Log Queries

Moderns Windows Event Logs are stored in a binary XML structure. Within the EventData XML element, there are numerous *<Data>* tags. For example, *<Data Name="LogonType">2</Data>* shows that the logon type is 2, meaning an interactive logon (as we described in our Event Log Analyst Reference). Using the Get-WinEvent cmdlet, we can filter on these specific data elements in order to retrieve only the event log records that match specific criteria which we set. To do so, we first create a query using an XPath expression and then call that query using the Get-WinEvent cmdlet with the -FilterXML parameter. To complete the first step, open a text editor and create a file named query.xml. In that file we will generate a query list with specific queries of interest to us. For example, the following query requests only Event ID 4624 where the target username is example_user:

```
<QueryList>
  <Query Id="0">
    <Select Path="Security">
       *[EventData[Data[@Name='TargetUserName'] and Data='example_user']]
       and
       *[System[(EventID=4624)]]</Select>
  </Query>
</QueryList>
```

To execute this query, once we have saved the query itself to the query.xml file, we run the following Get-WinEvent command:

```
Get-WinEvent -FilterXml ([xml](Get-Content .\query.xml))
```