PowerShell Remoting Cheatsheet

I have become a big fan of PowerShell Remoting. I find my self using it for both penetration testing and standard management tasks. In this blog I'll share a basic PowerShell Remoting cheatsheet so you can too.

Introduction to PowerShell Remoting

PowerShell Remoting is essentially a native Windows remote command execution feature that's build on top of the Windows Remote Management (WinRM) protocol. Based on my super Google results, WinRM is supported by Windows Vista with Service Pack 1 or later, Windows 7, Windows Server 2008, and Windows Server 2012.

Enabling PowerShell Remoting

Before we get started let's make sure PowerShell Remoting is all setup on your system.

1. In a PowerShell console running as administrator enable PowerShell Remoting.

   ```
   Enable-PSRemoting –force
   ```

   This should be enough, but if you have to troubleshoot you can use the commands below

2. Make sure the WinRM service is setup to start automatically.

   ```
   # Set start mode to automatic
   Set-Service WinRM -StartMode Automatic

   # Verify start mode and state - it should be running
   Get-WmiObject -Class win32_service | Where-Object {$_.name -like "WinRM"}
   ```

3. Set all remote hosts to trusted. Note: You may want to unset this later.

   ```
   # Trust all hosts
   Set-Item WSMan:localhost\client\trustedhosts -value *

   # Verify trusted hosts configuration
   Get-Item WSMan:\localhost\Client\TrustedHosts
   ```

Executing Remote Commands with PowerShell Remoting

Now we can play around a little. There's a great blog from a while back that provides a nice overview of PowerShell Remoting at http://blogs.technet.com/b/heyscriptingguy/archive/2009/10/29/hey-scripting-guy-october-29-2009.aspx. It's definitely on my recommended reading list, but I'll expand on the examples a little.

- ## Executing a Single Command on a Remote System

  The “Invoke-Command” command can be used to run commands on remote systems. It can run as the current user or using alternative credentials from a non domain system. Examples below.

  ```
  Invoke-Command –ComputerName MyServer1 -ScriptBlock {Hostname}
  Invoke-Command –ComputerName MyServer1 -Credential demo\serveradmin -
  ScriptBlock {Hostname}
  ```

  If the ActiveDirectory PowerShell module is installed it’s possible to execute commands on many systems very quickly using the pipeline. Below is a basic example.

  ```
  Get-ADComputer -Filter *  -properties name | select
  @{Name="computername";Expression={$_."name"}} | Invoke-Command -ScriptBlock
  {hostname}
  ```

  Sometimes it’s nice to run scripts stored locally on your system against remote systems. Below are a few basic examples.

  ```
  Invoke-Command -ComputerName MyServer1 -FilePath C:\pentest\Invoke-
  Mimikatz.ps1
  Invoke-Command -ComputerName MyServer1 -FilePath C:\pentest\Invoke-
  Mimikatz.ps1 -Credential demo\serveradmin
  ```

  Also, if your dynamically generating commands or functions being passed to remote systems you can use invoke-expression through invoke-command as shown below.

  ```
  $MyCommand = "hostname"
  $MyFunction = "function evil {write-host `"Getting evil...`";iex -command
  $MyCommand};evil"
  invoke-command -ComputerName MyServer1 -Credential demo\serveradmin -
  ScriptBlock {Invoke-Expression -Command  "$args"} -ArgumentList $MyFunction
  ```

- ## Establishing an Interactive PowerShell Console on a Remote System

  An interactive PowerShell console can be obtained on a remote system using the “Enter-PsSession” command. It feels a little like SSH. Similar to “Invoke-Command”, “Enter-PsSession” can be run as the current user or using alternative credentials from a non domain system. Examples below.

  ```
  Enter-PsSession –ComputerName server1.domain.com
  Enter-PsSession –ComputerName server1.domain.com –Credentials
  domain\serveradmin
  ```

  If you want out of the PowerShell session the “Exit-PsSession” command can be used.

  ```
  Exit-PsSession
  ```

- ## Creating Background Sessions

  There is another cool feature of PowerShell Remoting that allows users to create background sessions using the “New-PsSession” command. Background sessions can come in handy if you want to execute multiple commands against many systems. Similar to the other commands, the “New-PsSession” command can run as the current user or using alternative credentials from a non domain system. Examples below.

  ```
  New-PSSession -ComputerName server1.domain.com
  New-PSSession –ComputerName server1.domain.com –Credentials
  domain\serveradmin
  ```

  If the ActiveDirectory PowerShell module is installed it’s possible to create background sessions for many systems at a time (However, this can be done in many ways). Below is a command example showing how to create background sessions for all of the domain systems. The example shows how to do this from a non domain system using alternative domain credentials.

  ```
  New-PSDrive -PSProvider ActiveDirectory -Name RemoteADS -Root "" -Server
  a.b.c.d -credential domain\user
  cd RemoteADS:
  Get-ADComputer -Filter * -Properties name  | select
  @{Name="ComputerName";Expression={$_."name"}} | New-PSSession
  ```

- ## Listing Background Sessions

  Once a few sessions have been established the “Get-PsSession” command can be used to view them.

  ```
  Get-PSSession
  ```

- ## Interacting with Background Sessions

  The first time I used this feature I felt like I was working with Metasploit sessions, but these sessions are a little more stable. Below is an example showing how to interact with an active session using the session id.

  ```
  Enter-PsSession –id 3
  ```

  To exit the session use the “Exit-PsSession” command. This will send the session into the background again.

  ```
  Exit-PsSession
  ```

- ## Executing Commands through Background Sessions

  If your goal is to execute a command on all active sessions the “Invoke-Command” and “Get-PsSession” commands can be used together. Below is an example.

```
Invoke-Command -Session (Get-PSSession) -ScriptBlock {Hostname}
```

- **Removing Background Sessions**

  Finally, to remove all of your active sessions the “Disconnect-PsSession” command can be used as shown below.

  ```
  Get-PSSession | Disconnect-PSSession
  ```

Wrap Up

Naturally PowerShell Remoting offers a lot of options for both administrators and penetration testers. Regardless of your use case I think it boils down to this:

- Use “Invoke-Command” if you’re only going to run one command against a system
- Use “Enter-PSSession” if you want to interact with a single system
- Use PowerShell sessions when you’re going to run multiple commands on multiple systems

Hopefully this cheatsheet will be useful. Have fun and hack responsibly.

References

- http://blogs.technet.com/b/heyscriptingguy/archive/2009/10/29/hey-scripting-guy-october-29-2009.aspx
- https://technet.microsoft.com/en-us/library/hh849694.aspx
- https://technet.microsoft.com/en-us/magazine/ff700227.aspx
- https://technet.microsoft.com/en-us/magazine/ff394367.aspx