PowerShell – You're Using It, Right?

- **Name:**
  Chris Thomas

- **Title:**
  Desktop Engineer

- **Organization:**
  Ingham Intermediate School District

- **Email:**
  cthomas@inghamisd.org

- **Twitter:**
  @AutomateMyStuff

- **Past Presentations:**
  https://github.com/chrisATautomatemystuff/Presentations

PowerShell – You're Using It, Right?

PowerShell has the ability to help save you time and effort while also providing you consistent results and documented processes. Find those annoying manual tasks you or your staff do and come learn how you can start to remove some of that manual effort for automated tasks.

We'll cover some fundamentals, but try to spend the majority of the time focusing on real-world examples of how we use PowerShell at our organization to make our lives easier.

Who Am I?

- 22+ Years In K12 Technology
  - Intern, Technician, Coordinator, Engineer
- Lifelong Learner
  - /r/sysadmin, /r/k12sysadmin, RSS feeds
- Relentlessly Inquisitive
  - Let's Ask The WinAdmins Community
- Problem Solver
  - Professional Googler
- Voracious Reader
  - learn.microsoft.com/en-us/docs/
- Community Minded
  - MAEDS, MISCUG, WMISMUG
- Past Presentations
  - github.com/chrisATautomatemystuff/Presentations
- #ImpostorSyndromeBeDamned

There's a struggle to it and that's normal

Don Jones: "If you're not willing to play a little bit you'll probably not be successful at PowerShell."

Jeffrey Snover: "I'm a Distinguished Engineer, I'm the Lead Architect with Windows Server and System Center Datacenter, and I invented the dang thing and still there's a struggle to it and that's normal.

Windows PowerShell Unplugged with Jeffrey Snover & Don Jones
TechEd North America 2014 • (1hr 16min • quotes @ 3:15)
https://www.youtube.com/watch?v=qVIPNsAkJxM
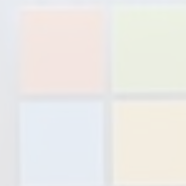
Shell: the command interpreter that is used to pass commands to the operating system

ISE: the Integrated Scripting Environment is an application where you can run, test, and debug scripts in a single GUI with tab completion, syntax coloring, selective execution, and context-sensitive help

Cmdlet: a task-oriented command that is typically used to return a .NET Framework object to the next command in the pipeline

Variable: a name given to stored information, e.g. $users, $iWishYouHadAStrongPassword, $x

Vocabulary 01

Parameter: input values or arguments used by the cmdlet or script to make it more dynamic

Named:

```
PS C:\>
PS C:\> Get-ChildItem -Path $env:USERPROFILE\AppData\Local\Temp -Filter *.exe
```

Positional:

```
PS C:\>
PS C:\> Get-ChildItem $env:USERPROFILE\AppData\Local\Temp *.exe
```

Switch:

```
PS C:\>
PS C:\> Get-ChildItem -Path $env:USERPROFILE\AppData\Local\Temp -Filter *.exe -Recurse
```

Object: a representation of something with methods to take actions against it and properties to access information stored within it

Vocabulary 02

Objects - The Bicycle Example

Property

Something about the object

- **Wheel Size**
- **Tire Pressure**
- **Height**
- **Gears**
- **Color**
- **Speed**
- **Price**

Method

Something you can do with or to the object

- **Pedal**
- **Brake**
- **Change Gear**
- **Repair**
- **Re-inflate Tire**

Array: a data structure that serves as a collection of multiple items. You can iterate over the array or access individual items using an index

Function: a command or series of commands grouped to run together

Module: a set of related Windows PowerShell functionalities, grouped together as a convenient unit (usually saved in a single directory)

Script Module: a file (.psm1) that contains PowerShell code for functions, variables and more

Binary Module: a .NET Framework assembly (.dll) that contains compiled code

Dynamic Module: a module that only exists in memory (Import-PSSession)

Vocabulary 03

Vocabulary 04

Pipeline (|): a method to send the results of the preceding command as input to the next command

$_: "the current object in the pipeline", or "this", or $PSItem

Alias: shortcut to a command, cmdlet or function

Comment: You can type a comment symbol (#) before each line of comments, or you can use the (<#) and(#>) symbols to create a comment block

Escape Character: the grave-accent (`) is a continuation character if used at end of line or displays literal value of a string / variable

Command History: <up arrow> or <down arrow> and Get-History

Vocabulary 05

A **inexperienced** and/or **ignorant** or **unskilled** person. Especially used in computer games.

Exploratory Learning For N00bs

- New-Item –Path C:\MAEDS
- Set-Location –Path C:\MAEDS

<SHOCKED PIKACHU>

- Get-ChildItem
- Remove-Item –Path C:\MAEDS
- New-Item –Path C:\MAEDS –ItemType Directory
- Set-Location –Path C:\MAEDS
- Get-ChildItem

In Information Technology and specifically Computer Science, the term "**Grey beard**" or "**Greybeard**" is used as a **droll**, sarcastic way to describe any person who is older. Typically, a **Grey Beard** in IT will be very specific about methodology and have well-reasoned arguments about why the methods they are using are superior to current ideas and methods. Many times, they're not wrong!

So me think, why waste time say lot word, when few word do trick.

Exploratory Learning For Greybeards

- mkdir c:\maeds
- cd c:\maeds
- dir
- cd..
- rm maeds

<PYSCH you just learned POSH>

- Get-Help mkdir
- Get-Help cd
- Get-Help dir
- Get-Help rm
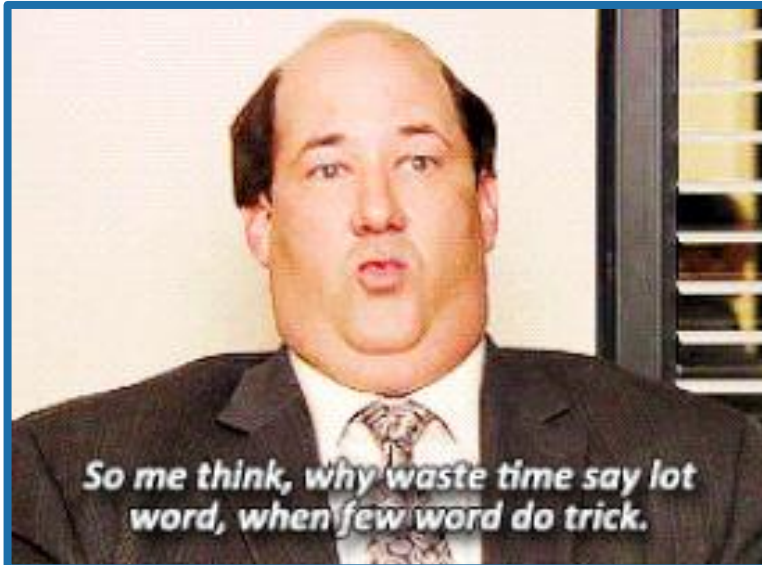
Get-Command

- Get-Command -Module *<moduleName>*
- Get-Command -Verb Get
- Get-Command -Verb Get -Noun P*
- Get-Command Get-P*
- Get-Command Get-P* -Module Microsoft.PowerShell.Utility
- Get-Command Get-P* -Module Microsoft.PowerShell.Management

Get-Help

- Get-Help *<cmdlet>* -Examples
- Get-Help *<cmdlet>* -Full
- Get-Help *<cmdlet>* -ShowWindow
- Get-Help *<cmdlet>* -Online
- Get-Help *<cmdlet>* | Clip

Get-Member

- *<cmdlet>* | Get-Member

- **Caveat**
- Get-ADUser cthomas | Get-Member

vs.

- Get-ADUser cthomas -Properties * | Get-Member
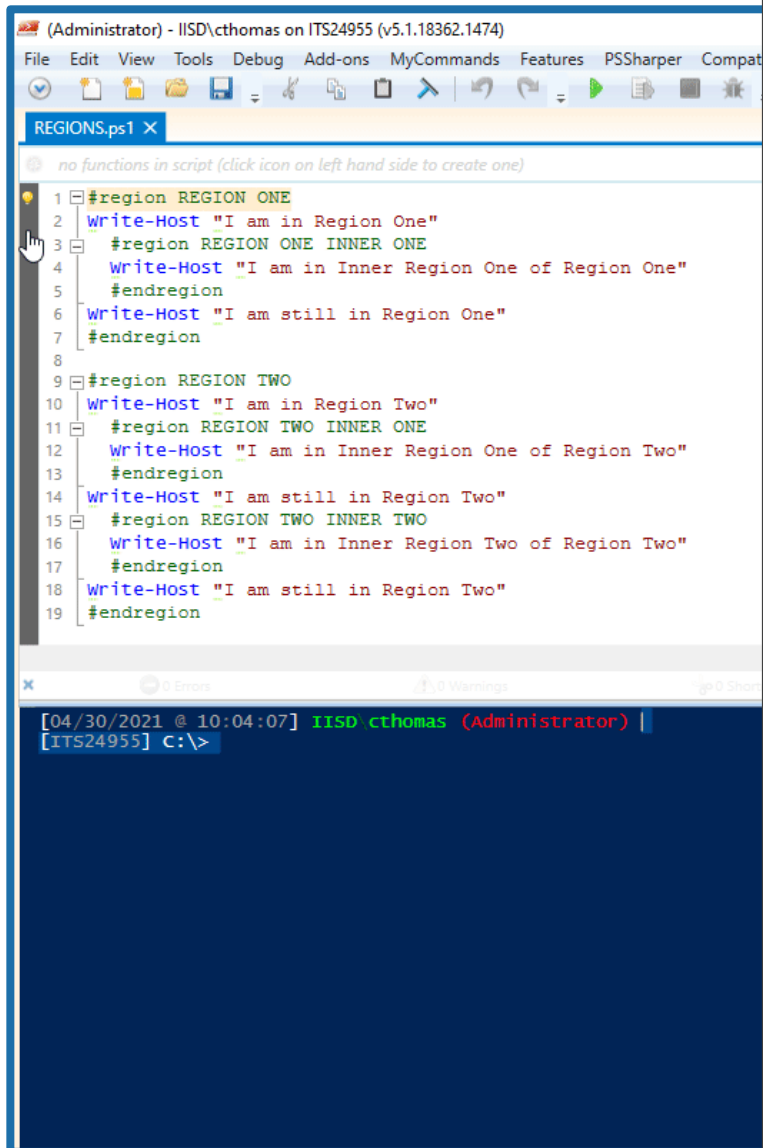
Get-Module

- Get-Module

- Get-Module –ListAvailable

- $session_DC1 = New-PSSession -ComputerName V-DC1
- Get-Module -PSSession $session_DC1 -ListAvailable

Tab Expansion

```
Administrator: Windows PowerShell
PS C:\>
```
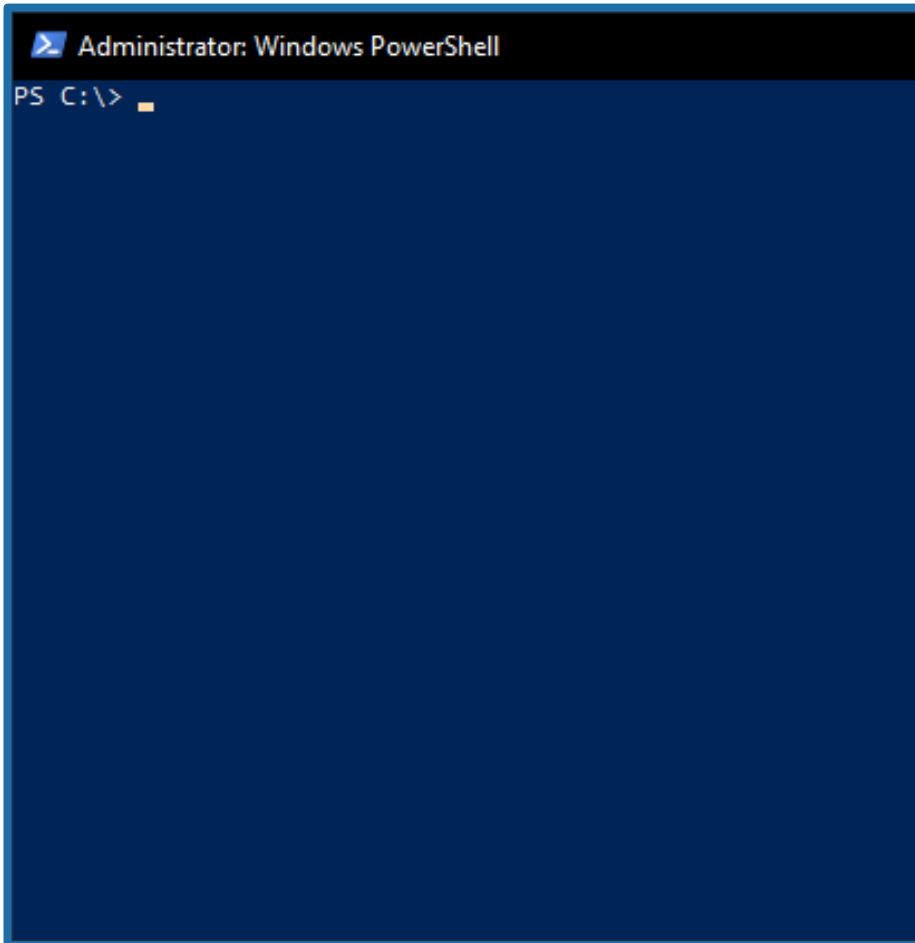
- new-adu<TAB>

- New-ADUser -<TAB>
  - <TAB> to advance through parameters
    - or
  - <SHIFT+TAB> to go back to a previous parameter

- New-ADUser -<CTRL + SPACE>
  - <ARROW KEYS> + <SPACE> to select appropriate parameter

Regions

- Separate code into collapsible sections

- <CTRL> + M to collapse/expand all regions

- Collapsing regions only affects readability, not runability

Command Syntax

```
SYNTAX
    New-ADUser [-Name] <String> [-AccountExpira
    [-AuthenticationPolicySilo <ADAuthenticatio
    <String>] [-CompoundIdentitySupported <Bool
    <String>] [-EmployeeNumber <String>] [-Enab
    [-KerberosEncryptionType {None | DES | RC4
    <Hashtable>] [-OtherName <String>] [-PassTh
    [-ProfilePath <String>] [-SamAccountName <S
    [-Title <String>] [-TrustedForDelegation <B
```

```
SYNTAX
    Add-ADGroupMember [-Identity] <ADGroup> [-Members] <ADPri
```

- <command-name> -<Required Parameter Name> <Required Parameter Value>

- [-<Optional Parameter Name> <Optional Parameter Value>]

- [-<Optional Switch Parameters>]

- [-<Optional Parameter Name>] <Required Parameter Value>

Splatting

```powershell
1   Import-Module ActiveDirectory
2
3   #region Import File Locations
4   $students = Import-Csv C:\scripts\students_import.csv
5   #endregion
6
7   foreach ($student in $students){
8
9       #region Student Variable Definitions
10          $FN = $student.FN
11          $LN = $student.LN
12          $NAME = $FN + " " + $LN
13          $YOG = $student.YOG
14          $SAM = $student.SN
15          $TEMPPWD = (ConvertTo-SecureString -AsPlainText "IWishYouHad8DigitPasswo
16          $INSECUREPWD = $student.PWD
17          $SECUREPWD = (ConvertTo-SecureString -AsPlainText $INSECUREPWD -Force)
18          $UPN = $SAM + "@<DOMAIN NAME>"
19          $EMAIL = $SAM + "@<DOMAIN NAME>"
20          $PATH = "OU=" + $YOG + ",OU=StudentAccounts,OU=UserAccounts,DC=<DOMAIN N
21          $GROUP = "U_All_" + $YOG + "Students"
22      #endregion
23
24      $newUserSplat = @{
25          Name = $NAME
26          DisplayName = $NAME
27          GivenName = $FN
28          Surname = $LN
29          SamAccountName = $SAM
30          UserPrincipalName = $UPN
31          EmailAddress = $EMAIL
32          AccountPassword = $TEMPPWD
33          CannotChangePassword = $true
34          PasswordNeverExpires = $true
35          Path = $PATH
36          PassThru = $null
37      }
38
39      #region Active Directory Student Provisioning
40          New-ADUser @newUserSplat | Enable-ADAccount
41
42          Add-ADGroupMember -Identity $GROUP -Members $SAM
43
44          Set-ADAccountPassword -Identity $SAM -Reset -NewPassword $SECUREPWD
45      #endregion
46  }
```

- Remove the need for horizontal scrolling

- Remove the need for backticks (or troubleshooting forgotten backticks…)

- Cleaner look to the script

PowerShell Profile

```
Microsoft.PowerShellISE_profile.ps1 ×

Roll-Rick

701         "Never gonna give you up, never gonna let you down",
702         "Never gonna run around and desert you",
703         "Never gonna make you cry, never gonna say goodbye",
704         "Never gonna tell a lie and hurt you"
705
706  □  Invoke-Command -ComputerName $ComputerName -ArgumentList (,$Rick)
707         Param ([String[]]$Rolling)
708         [Reflection.Assembly]::LoadWithPartialName('System.Speech') |
709         $Speech = New-Object System.Speech.Synthesis.SpeechSynthesizer
710         ForEach ($Verse in $Rolling)
711  □      {
712             $Command = {msg * "$Verse"}
713             $Command | Invoke-Expression
714             $Speech.Speak($Verse)
715             Sleep 1
716         }
717     }
718  }
719
720  #requires -Module ActiveDirectory
721  #Import-Module ActiveDirectory -EA Stop
722
723 ⊞Function Get-AccountLockoutStatus {...}
836
837 ⊞function QueryRdpConnections {...}
865
866  #endregion
867
868 □#region PSDrives
869  New-PSDrive -Name sccm -PSProvider FileSystem -Root '\\
870  New-PSDrive -Name staging -PSProvider FileSystem -Root '\\
871  #endregion
872
873  #cleanup my o365 sessions
874 ⊞Register-EngineEvent PowerShell.Exiting -SupportEvent -Action {...}
881
882  #TAKE ME HOME
883  Set-Location 'C:\'
884
885  #GET OUT OF MY FACE
886  Clear-Host
887
888 □$intro = @"
889
890
891
892
893                          DON'T BREAK STUFF!
894
895
896          L__;__J
897
898
```

- If you don't have a $profile, create one and get going!

  - if (! (Test-Path $Profile)) { New-Item -Type File -Path $Profile -Force }
  - powershell_ise $profile

```
  1 ⊞$sqlQuery = @"..."@
189
190   $inven = Invoke-Sqlcmd -ServerInstance "                    " -Database "      " -Query $sqlQuery
191
192   $inven | Export-Csv -Path C:\scripts\devices_w_primary_users_$(Get-Date -Format MMddyy).csv -NoTypeInformation
```

Name	ADSite	User Name
EO		
EO		
EO		
EO		
EO		
ER		
ER		
ER		
ER		
ER		
ER		
ER		
ER		
ER		
GR		
GR		
GR		
GR		
GR		
GR		

Model	Serial Number	BIOSDate
Latitude 3520	FPCBRG3	9/12/2021 0:0
Latitude 5490	BRWKQN2	8/27/2018 0:0
Latitude 7490	3RLPHR2	7/12/2018 0:0
Latitude 7420	DL6G6G3	7/12/2022 0:0
Latitude 7275	FBX2M72	7/9/2016 0:0
Latitude 7275	FBX2M72	7/9/2016 0:0
Latitude E7470	G2CVKC2	8/20/2018 0:0
Latitude E7470	G2CVKC2	8/20/2018 0:0
OptiPlex 7060	DD6M7X2	11/1/2018 0:0
OptiPlex 7060	DF0M7X2	11/1/2018 0:0
OptiPlex 7060	DF3Q7X2	11/1/2018 0:0
Latitude 3410	B1T8303	7/17/2020 0:0
HP EliteDesk 800 G1 SFF	MXL4291XSZ	5/9/2017 0:0
Latitude E6440	DSGQQ72	10/9/2018 0:0
OptiPlex 7060	DCNN7X2	11/1/2018 0:0
OptiPlex 7060	DDMK7X2	11/1/2018 0:0
Latitude 5520	5B7F1B3	7/4/2022 0:0
HP EliteDesk 800 G1 SFF	MXL4291XNG	12/17/2013 0:0
HP EliteDesk 800 G1 SFF	MXL4291XQN	12/17/2013 0:0

Last Reboot Date/Time	Install Date	Last Hardware Inventory
Oct 3 2022 9:53AM	8/8/2022	10/3/2022
Oct 18 2022 7:44AM	7/5/2022	10/18/2022
Oct 18 2022 11:03AM	6/30/2022	10/18/2022
Oct 12 2022 1:35PM	6/30/2022	10/18/2022
Oct 3 2022 9:38AM	8/1/2022	10/18/2022
Oct 3 2022 9:38AM	8/1/2022	10/18/2022
Oct 17 2022 8:02AM	12/9/2021	10/17/2022
Oct 17 2022 8:02AM	12/9/2021	10/17/2022
Oct 17 2022 7:43AM	1/8/2022	10/18/2022
Oct 3 2022 7:54AM	1/21/2022	10/17/2022
Oct 18 2022 10:37AM	2/25/2022	10/18/2022
Oct 6 2022 2:40PM	9/30/2022	10/6/2022
Oct 3 2022 8:15AM	7/19/2022	10/17/2022
Sep 26 2022 12:44PM	7/25/2022	9/26/2022
Oct 17 2022 7:03AM	1/14/2022	10/18/2022
Oct 17 2022 7:31AM	8/9/2022	10/18/2022
Oct 8 2022 3:29PM	4/22/2021	10/18/2022
Sep 29 2022 1:12PM	6/29/2020	10/17/2022
Oct 10 2022 10:23AM	8/23/2019	10/18/2022
Oct 3 2022 7:35AM	1/7/2022	10/18/2022
Oct 12 2022 8:23AM	1/10/2022	10/18/2022
Oct 3 2022 8:20AM	1/7/2022	10/18/2022
Oct 17 2022 8:23AM	7/25/2022	10/18/2022

Windows Updates Scan Tim	Windows Updates Last Error Cod
8/22/2022 18:41	-2016410110
8/30/2021 14:35	-2147467259
	-2147467259
2/25/2022 17:21	-2147467259
4/26/2021 11:06	-2147467259
4/26/2021 11:06	-2147467259
10/2/2022 18:01	-2147023890
10/2/2022 18:01	-2147023890
3/29/2021 14:05	-2147467259
4/12/2021 12:09	-2147467259
3/18/2021 16:26	-2147467259
10/4/2022 16:08	-2145123264
11/17/2020 18:23	-2147467259
9/26/2022 17:05	-939523078
	-2147467259
	-2147467259
3/11/2021 20:14	-2147467259
	-2147467259
	-2147467259
	-2147467259
	-2147467259
9/13/2021 16:14	-2147467259
	-2147467259

Hardware Inventory

SQL Queries

Invoke-Sqlcmd

Export-Csv

```
references | Pester
function New-HomeFolder ($homeDrivePath, $samAccountName)
{
    $domainName = "iisd"

    #CREATE HOME DIRECTORY
    New-Item -Path $homeDrivePath -ItemType Directory -Force

    #APPLY PERMISSIONS TO HOME FOLDER
    $identityReference = $domainName + '\' + $samAccountName
    $fileSystemAccessRights = [System.Security.AccessControl.FileSystemRights]::Modify
    $inheritanceFlags = [System.Security.AccessControl.InheritanceFlags]::ContainerInherit -bor [System.Security.AccessControl.InheritanceFlags]::ObjectInherit
    $propagationFlags = [System.Security.AccessControl.PropagationFlags]::None
    $accessControl = [System.Security.AccessControl.AccessControlType]::Allow
    $accessRuleSplat = $identityReference, $fileSystemAccessRights, $inheritanceFlags, $propagationFlags, $accessControl
    $accessRule = New-Object System.Security.AccessControl.FileSystemAccessRule $accessRuleSplat
    $homeDirectoryACL = Get-Acl $homeDrivePath
    $homeDirectoryACL.AddAccessRule($accessRule)
    Set-Acl -Path $homeDrivePath -AclObject $homeDirectoryACL
}
```

Managing Home Drives

New-Item

Get-Acl

Set-Acl

PowerShell App Deployment Toolkit (PSADT)

PowerShell App Deployment Toolkit Variables

Toolkit Name
$appDeployToolkitName
$appDeployMainScriptFriendlyName
Script Info
$appDeployMainScriptVersion
$appDeployMainScriptMinimumConfigVersion
$appDeployMainScriptDate
$appDeployMainScriptParameters

Datetime and Culture
$currentDateTime
$currentTime
$currentDate
$currentTimeZoneBias
$culture
$currentLanguage
$currentUILanguage

Environment Variables
$envHost
$envShellFolders
$envAllUsersProfile
$envAppData
$envArchitecture
$envCommonProgramFiles
$envCommonProgramFilesX86
$envCommonDesktop
$envCommonDocuments
$envCommonStartMenuPrograms
$envCommonStartMenu
$envCommonStartUp
$envCommonTemplates
$envComputerName
$envComputerNameFQDN
$envHomeDrive
$envHomePath

$envHomeShare
$envLocalAppData
$envLogicalDrives
$envProgramFiles
$envProgramFilesX86
$envProgramData
$envPublic
$envSystemDrive
$envSystemRAM
$envSystemRoot
$envTemp
$envUserCookies
$envUserDesktop
$envUserFavorites
$envUserInternetCache
$envUserInternetHistory
$envUserMyDocuments
$envUserName
$envUserProfile
$envUserSendTo
$envUserStartMenu
$envUserStartMenuPrograms
$envUserStartUp
$envSystem32Directory
$envWinDir

Domain Membership
$IsMachinePartOfDomain
$envMachineWorkgroup
$envMachineADDomain
$MachineDomainController
$envMachineDNSDomain
$envUserDNSDomain
$envUserDomain

Operating System
$envOS
$envOSName
$envOSServicePack
$envOSVersion
$envOSVersionMajor
$envOSVersionMinor
$envOSVersionBuild
$envOSVersionRevision
$envOSProductType
$IsServerOS
$IsDomainControllerOS
$IsWorkStationOS
$envOSProductTypeName
$Is64Bit
$envOSArchitecture

Current Process Architecture
$Is64BitProcess
$psArchitecture

PowerShell And CLR (.NET) Versions
$envPSVersionTable
$envPSVersion
$envPSVersionMajor
$envPSVersionMinor
$envPSVersionBuild
$envPSVersionRevision
$envCLRVersion
$envCLRVersionMajor
$envCLRVersionMinor
$envCLRVersionBuild
$envCLRVersionRevision

Permissions/Accounts
$CurrentProcessToken
$CurrentProcessSID

$ProcessNTAccount
$ProcessNTAccountSID
$IsAdmin
$IsLocalSystemAccount
$IsLocalServiceAccount
$IsNetworkServiceAccount
$IsServiceAccount
$IsProcessUserInteractive
$LocalSystemNTAccount
$SessionZero

Script Name and Script Paths
$scriptPath
$scriptName
$scriptFileName
$scriptRoot
$invokingScript
$scriptParentPath

App Deploy Script Dependency Files
$appDeployLogoIcon
$appDeployLogoBanner
$appDeployConfigFile
$appDeployToolkitDotSourceExtensions
$xmlConfigFile
$configConfigVersion
$configConfigDate

Script Directories
$dirFiles
$dirSupportFiles
$dirAppDeployTemp

Script Naming Convention
$appVendor
$appName

$appVersion
$appLang
$appRevision
$appArch
$installTitle
$installName

Executables
$exeWusa
$exeMsiexec
$exeSchTasks

RegEx Patterns
$MSIProductCodeRegExPattern

Registry Keys
$regKeyApplications
$regKeyLotusNotes
$regKeyAppExecution
$regKeyDeferHistory

COM Objects
$Shell
$ShellApp

Log File
$logName
$logTempFolder
$configToolkitLogDir
$DisableScriptLogging
$RevertScriptLogging

Script Parameters
$deployAppScriptParameters
$appDeployMainScriptParameters
$appDeployExtScriptParameters

Logged On Users
$LoggedOnUserSessions
$usersLoggedOn
$CurrentLoggedOnUserSession
$CurrentConsoleUserSession
$RunAsActiveUser

Miscellaneous
$dpiPixels
$runningTaskSequence
$IsTaskSchedulerHealthy
$invalidFileNameChars
$useDefaultMsi

Mailbox Forwarding / Inbox Rules Auditing

- Get-Mailbox -Identity cthomas@inghamisd.org | Select-Object -Property *forward*

- Get-InboxRule -Mailbox cthomas@inghamisd.org -IncludeHidden | Select-Object -Property name,description,enabled,redirectto,movetofolder,forwardto | Format-List

- Get-InboxRule -Mailbox cthomas@inghamisd.org -IncludeHidden | Select-Object -Property name,description,enabled,redirectto,movetofolder,forwardto | Where-Object -FilterScript {$_.redirectto -ne $null -or $_.forwardto -ne $null}

Mailbox Size Audits & Modifications

- Get-MailboxFolderStatistics -Identity cthomas@inghamisd.org -FolderScope Inbox | Select-Object -Property name,foldersize,itemsinfolder

- Get-Mailbox -Identity cthomas@inghamisd.org | Select-Object -Property *quota*

- Set-Mailbox -Identity cthomas@inghamisd.org -IssueWarningQuota 90GB -ProhibitSendQuota 95GB -ProhibitSendReceiveQuota 99GB
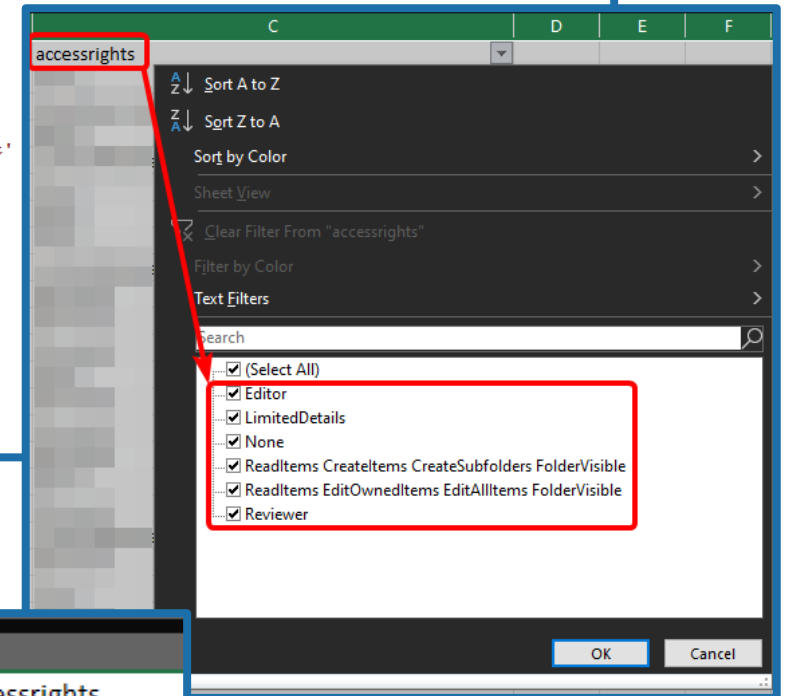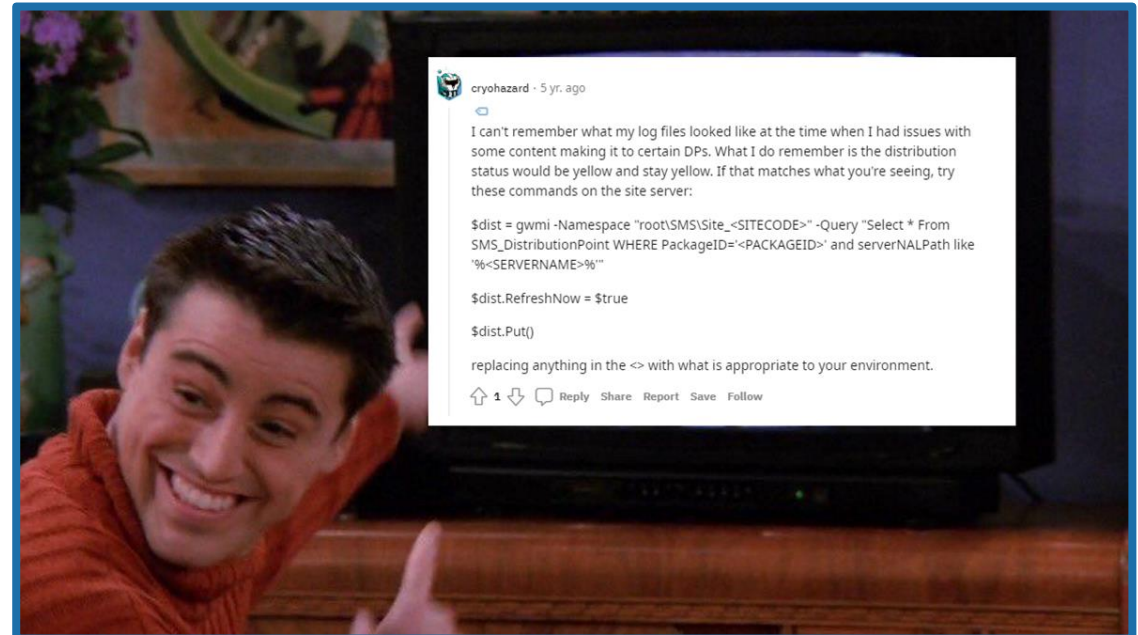
Calendar Permission Audit

ConfigMgr – Content Issues

$dist = gwmi -Namespace "root\SMS\Site_<SITECODE>" -Query "SELECT * FROM SMS_DistributionPoint WHERE PackageID='<PACKAGEID>' AND serverNALPath LIKE '%<SERVERNAME>%'"

$dist.RefreshNow = $true

$dist.Put()

ConfigMgr – DP Issues Post IPU

- https://smsagent.blog/2020/10/19/real-world-notes-in-place-os-upgrade-on-server-2012-r2-configmgr-distribution-points/

Anyway, I'm documenting here what I did to fix it and the scripts I used.

To fix the broken ConfigMgr client I did the following:

1. Compile the mof file %program files%\Microsoft Policy Platform\ExtendedStatus.mof
2. Stop any ccmrepair.exe or ccmsetup.exe process so they don't hinder step 3
3. Run the 'Configuration Manager Health Evaluation' scheduled task (ccmeval.exe) a couple of times. This will self-remediate the WMI issues.

To fix the broken DP role:

1. Compile the mof file ..\SMS_DP$\sms\bin\smsdpprov.mof (this restores the missing WMI namespace)
2. Query the ConfigMgr database to find the list of packages distributed to the distribution point
3. Run some PowerShell code to restore the packages as instances in the root\SCCMDP:SMS_PackagesInContLib WMI class
4. Run the Content Validation scheduled task to revalidate the content and remove any errors in the console

For the latter, I don't take any credit in my script below as I simply used and expanded something I found here. Note both scripts must be run as administrator and the second script requires read access to the ConfigMgr database.

ConfigMgr/M365 Licensing Reconcilation

Print Server Management

- Invoke-Command -ComputerName <SERVERNAME> -ScriptBlock {Stop-Service -Name Spooler -Force}

- Invoke-Command -ComputerName <SERVERNAME> -ScriptBlock {Remove-Item C:\Windows\System32\spool\PRINTERS*.*}

- Invoke-Command -ComputerName <SERVERNAME> -ScriptBlock {Start-Service -Name Spooler}

<TOO MUCH TO REMEMBER!>

- function Reset-Spooler
- {
-     Param([Parameter(Mandatory=$true)]$PrintServerFQDN)
-     Invoke-Command -ComputerName $PrintServerFQDN -ScriptBlock {Stop-Service -Name Spooler -Force}
-     Invoke-Command -ComputerName $PrintServerFQDN -ScriptBlock {Remove-Item C:\Windows\System32\spool\PRINTERS*.*}
-     Invoke-Command -ComputerName $PrintServerFQDN -ScriptBlock {Start-Service -Name Spooler}
- }

LAPS

- Get-ADComputer –Identity (hostname)
- Get-ADComputer –Identity (hostname) | Select-Object –Property *

<HRMPH>

- Get-ADComputer –Identity (hostname) -Properties ms-mcs-admpwd
- Get-ADComputer –Identity (hostname) – Properties ms-mcs-admpwd | Select-Object –Property ms-mcs-admpwd | Clip

<WUT?!>

- Get-ADComputer –Identity (hostname) –Properties ms-mcs-admpwd | Select-Object –ExpandProperty ms-mcs-admpwd | Clip

PowerShell As 2FA Bypass Vector

- https://cybergladius.com/secure-windows-remote-management-from-2fa-bypass/

Wrap-Up: TLDR

Bypassing 2FA on Windows systems can be done quickly using built-in tools. Our job as Blue Team Defenders is to understand these attacks and build a defense against them. We identified four critical pieces of the WinRM attack and how we can mitigate their risk. Those mitigations are as follows.

1. Limit network access to servers that use WinRM on ports 5985 & 5986 through VLAN Segmentation with ACL rules and Windows Firewalls.
2. Limit password re-use of the local Administrator account.
3. Protect Domain Admin account credentialsstored in the system cache.
4. Block Basic User accounts from accessing PowerShell on their systems.

These defenses combined will significantly impede an Attacker from using WinRM against you.
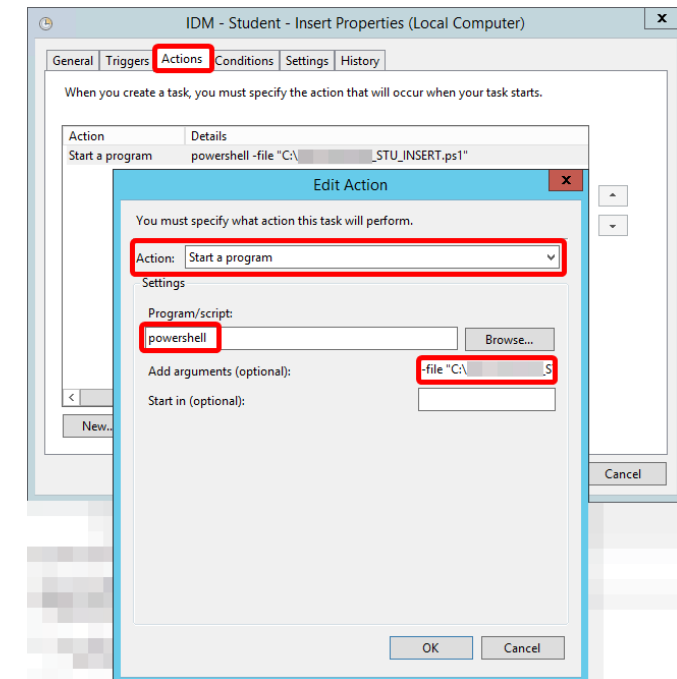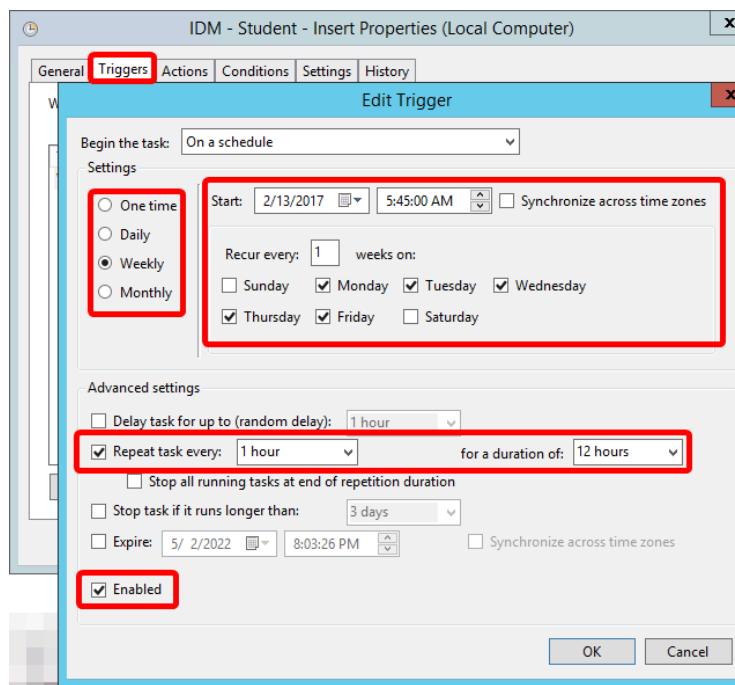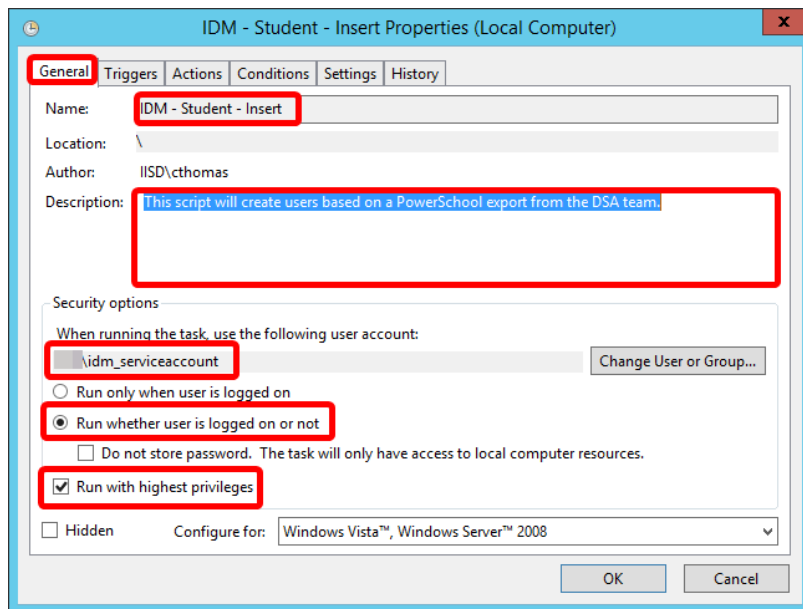
Task Scheduler

Run scripts on-demand until you are comfortable and then schedule them to run automatically so you can focus on other tasks

How to Schedule PowerShell Script using Task Scheduler

Use a service account with delegated permissions, not Domain Admin, to run your tasks

Active Directory Delegated Permissions Best Practices

Delegate AD group management

-- The End --

Ask Questions …

… then go forth and iterate.

PAST PRESENTATIONS:
https://github.com/chrisATautomatemystuff/Presentations