Assignment 7

Courtney Hagen

W0263284

OSYS 1200

Contents

[Introduction 3](#_Toc531011187)

[NewObject.ps1 3](#_Toc531011188)

[AssignACLs.ps1 3](#_Toc531011189)

[Backup.ps1 3](#_Toc531011190)

[PowerShell Profiles 4](#_Toc531011191)

[Execution Policies 5](#_Toc531011192)

[Change Management Log 7](#_Toc531011193)

[Change Management Log Attachments 8](#_Toc531011194)

[Change Management Log Changes Log Text 8](#_Toc531011195)

[References 9](#_Toc531011196)

# Introduction

The purpose of this document is to show the commands I used in PowerShell to make modifications to the virtual machine.

# NewObject.ps1

# Creating user Max

Net user Max student /add

# Making new group ITAdminsUsers

New-LocalGroup ITAdminsUsers -Description "Tech Support Admins"

#Adding ITAdmins to Administrators

net localgroup Administrators ITAdminsUsers /add

#Adding Max to ITAdminsUsers

net localgroup ITAdminsUsers Max /add

# Create a folder called "Backup" in root of C https://blogs.technet.microsoft.com/heyscriptingguy/2012/02/21/learn-four-ways-to-use-powershell-to-create-folders/

New-Item -Path c:\Backup -ItemType directory

# Creating subfolder called Profiles in this folder

New-Item -Path c:\Backup\Profiles -ItemType directory

# Creating folder on root of F called Files\_For\_Max

New-Item -Path f:\Files\_for\_Max -ItemType directory

# Copies all of the .bat files from the c:\Scripts folder to f:\Files\_for\_Max https://blogs.technet.microsoft.com/heyscriptingguy/2013/04/19/use-powershell-to-copy-files-and-folders-to-a-new-location/

Copy-Item -Path C:\Scripts\NewUser.bat -Destination f:\Files\_for\_Max

# AssignACLs.ps1

#Give Max modify privileges for F:\Files\_for\_Max\ referring to student examples provided.

#Creates acl variable for the path

$acl = get-acl F:\Files\_for\_Max

#Creates variable for the permissions for who and what

$permission = "Max", "Modify", "Allow"

#Sets variable for the access rule

$accessRule = New-object system.security.AccessControl.FileSystemAccessRule $permission

#Sets the acl variable to the created access rule variable

$acl.SetAccessRule($accessRule)

# uses this variable, using the other variables, to set the acl for the path

$acl | Set-acl F:\Files\_for\_Max

#Do the same thing with c:\backup\

$acl2 = get-acl C:\Backup

$permission2 = 'ITAdminsUsers', 'FullControl', 'Allow'

$accessRule2 = New-object system.security.AccessControl.FileSystemAccessRule $permission2

$acl2.SetAccessRule($accessRule2)

$acl2 | Set-acl C:\Backup

# Backup.ps1

# This gets the user of the environment executing this ps1 file

$cname = gc env:username

# This gets the date of when this ps1 file is being executed

$date = get-date -Format d.MMM.yyyy

# This creates the variable for the first path being backed up and when

$drive = "C:\Backup\$date"

# This creates the second path being backed up and when

$drive2 = "C:\Backup\Profiles\$date"

#This creates a variable for the first destination which is the path and the current user

$dest1 = $drive + $cname

# This creates the first backup directory in the destination path

new-item -path $dest1 -itemtype directory

#This creates a variable for the second destination which is the second path and the current user

$dest2 = $Drive2 + $cname

# This creates the second backup directory in the second destination path

new-item -path $dest2 -itemtype directory

# This creates a variable of $path F:\Files\_for\_Max"

$path = "F:\Files\_for\_Max"

# creates variable to include all items in the path with any file extension

$include = @("\*.\*")

#Gets all items in the path and recursevely all of child items, add hidden files, if a command fails, silently

#continue, for each item, copy the item to backup with recurse again and container preserves folder structure.

Get-ChildItem -path $path -Include $include -Recurse -force -erroraction SilentlyContinue | foreach { copy-item -path $\_ -Destination $dest1 -recurse -container } -erroraction SilentlyContinue

# Tis does similar to the previous include, except only uses file extensions .url and .msc

$include2 = @("\*.url\*","\*.msc\*")

# creates the first path to users

$path1 = "C:\Users\"

#creates the second path which is the first path and the name of the current user

$path2 = $path1 + $cname

$path

# This does the same as the previous but no error action in the copying of files and using the new variables.

Get-ChildItem -path $path2 -Include $include2 -Recurse -Force -erroraction SilentlyContinue | foreach { copy-item -path $\_ -Destination $dest2 -recurse -container } #-erroraction SilentlyContinue

#https://stackoverflow.com/questions/129088/what-is-the-meaning-of-powershells-copy-items-container-argument

#https://blogs.technet.microsoft.com/heyscriptingguy/2014/07/09/handling-errors-the-powershell-way/

#https://docs.microsoft.com/en-us/powershell/module/microsoft.powershell.management/get-childitem?view=powershell-6

#https://stackoverflow.com/questions/28732025/what-does-mean-in-powershell

# PowerShell Profiles

|  |  |  |
| --- | --- | --- |
| PowerShell Profile | Path | Description |
| Current User, Current Host -console | $Home\[My]Documents\WindowsPowerShell\Profile.ps1 | Configures console to the current user and host |
| Current User, All Hosts | $Home\[My]Documents\Profile.ps1 | Configures console to current user, but all hosts |
| All Users, Current Host -console | $PsHome\Microsoft.PowerShell\_profile.sp1 | Configures console to all users, but current host |
| All Users, All Hosts | $PsHome\Profile.ps1 | Configures console to all users and all hosts |
| Current user, Current Host -ISE | $Home\[My]Documents\WindowsPowerShell\Microsoft.PowerShellSE\_profile.ps1 | Configures console to current user and current host but in the ISE environment |
| All users, Current Host – ISE | $PsHome\Microsoft.PowerShellISE\_profile.ps1 | Configures console to current host and all users but in the ISE enviornment |

<https://blogs.technet.microsoft.com/heyscriptingguy/2012/05/21/understanding-the-six-powershell-profiles/>

https://4sysops.com/archives/the-powershell-profile/

# Execution Policies

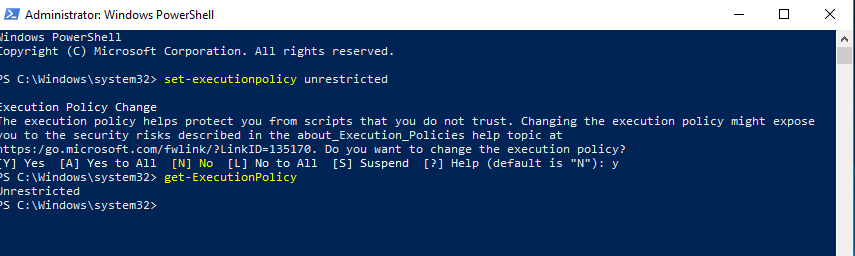
|  |  |
| --- | --- |
| PowerShell Execution Policy | Description |
| Restricted | Allows individual commands but not scripts to be run |
| AllSigned | Scripts can run but requires them to be signed by a trusted publisher (along with configuration files). Prompts before running scripts not classified as trusted/untrusted. Runs signed but malicious scripts |
| RemoteSigned | Scripts can run, and only requires a digital signature from a trusted publisher when downloaded from the internet, not the ones written on the local computer. It will, however, run the scripts downloaded and unsigned if unblocked by a command. Will also still run signed but malicious scripts. |
| Unrestricted | Unsigned scripts run, but warns user before running them if they’re downloaded from the internet. |
| Bypass | Runs everything and provides no warnings. Usually used when what it is being used for has its’ own security. |
| Undefined | No execution policy, uses current scope. If all scopes are undefined, will be set as restricted. |

https://docs.microsoft.com/en-us/powershell/module/microsoft.powershell.core/about/about\_execution\_policies?view=powershell-6

# Change Management Log



## Change Management Log Attachments







## Change Management Log Changes Log Text

Created PS1 file titled "NewObject.ps1 in the Scripts folder in C:\ which created a new user 'Max', a new group ITAdminsUsers, added Max to that new group, and added the new group to the Administrators group. This .ps1 file also created a folder titled "Backup" in C:\, and a subfolder in Backup titled "Profiles". It also created a folder called "Files\_for\_Max" on the root of the F: drive. Finally, it copies all of the .bat files from the c:\scripts folder into the new f:\Files\_for\_Max.

Created another PS1 file titled "AssignACLs.ps1 in the scripts folder. This .ps1 file modifies the f:\Files\_for\_Max ACLs to allow Max modify priveleges. It also modifies the ACLs to allow ITAdmins full control of c:\Backup

Created Backup.ps1 in the Scripts folder which creates backups for the current user at the current time. Created a copy of this in the Backups folder but only allowed Max access to it. Set the execution policy to unrestricted.

# References

<https://blogs.technet.microsoft.com/heyscriptingguy/2012/02/21/learn-four-ways-to-use-powershell-to-create-folders/>

<https://blogs.technet.microsoft.com/heyscriptingguy/2013/04/19/use-powershell-to-copy-files-and-folders-to-a-new-location/>

<https://stackoverflow.com/questions/129088/what-is-the-meaning-of-powershells-copy-items-container-argument>

<https://blogs.technet.microsoft.com/heyscriptingguy/2014/07/09/handling-errors-the-powershell-way/>

https://docs.microsoft.com/en-us/powershell/module/microsoft.powershell.management/get-childitem?view=powershell-6

#https://stackoverflow.com/questions/28732025/what-does-mean-in-powershell

<https://docs.microsoft.com/en-us/powershell/module/microsoft.powershell.core/about/about_execution_policies?view=powershell-6>

<https://blogs.technet.microsoft.com/heyscriptingguy/2012/05/21/understanding-the-six-powershell-profiles/>

https://4sysops.com/archives/the-powershell-profile/