-- SQLServer\_SQLServer\_Code\_PowerShell - New.ps1

$strWMI = "\\" + $fileServer + "\root\cimv2:win32\_LogicalShareSecuritySetting.Name='" + $share.name + "'"

$objWMI\_ThisShareSec = [wmi]$strWMI

$shareSec = gwmi Win32\_LogicalShareSecuritySetting -filter "name='$($share.name)'"

if($shareSec) {

$sd = $shareSec.invokeMethod('GetSecurityDescriptor',$null,$null)

$ShareInfo += $sd.Descriptor.DACL

----------------------------------------------------------------------------------------------------------------------------------------------------------------

-- SQLServer\_SQLServer\_Code\_PowerShell - ListofUsers.ps1

function Listofusers

{

$strFilter = “(&(objectCategory=User))”

$objDomain = New-Object System.DirectoryServices.DirectorySearcher

$objSearcher = New-Object System.DirectoryServices.DirectorySearcher

$ObjSearcher.SearchRoot = $ObjDomain

$ObjSearcher.PageSize = 10000

$objSearcher.Filter = $strFilter

$objSearcher.SearchRoot = "dc=domain,dc=local"

$colResults = $objSearcher.FindAll()

foreach($objResult in $colResults)

{

$objItem = $objResult.Properties; $objItem.name

}

}

----------------------------------------------------------------------------------------------------------------------------------------------------------------

-- SQLServer\_SQLServer\_Code\_PowerShell - List of users.ps1

function Listofusers

{

$strFilter = "(&(objectCategory=User))"

$objDomain = New-Object System.DirectoryServices.DirectorySearcher

$objSearcher = New-Object System.DirectoryServices.DirectorySearcher

$ObjSearcher.SearchRoot = $ObjDomain

$ObjSearcher.PageSize = 10000

$objSearcher.Filter = $strFilter

$objSearcher.SearchRoot = "dc=domain,dc=local"

$colResults = $objSearcher.FindAll()

foreach($objResult in $colResults)

{

$objItem = $objResult.Properties; $objItem.name

}

}

ListofUsers

----------------------------------------------------------------------------------------------------------------------------------------------------------------

-- Powershell scripts\_users.ps1

# Start of script

cls

foreach ($targetou in 'A','B','C','D','E','F','G','GUESTACCOUNTS','H','I','J','K','L','CONTRACTOR','M','N','O','P','Q','R','S','T',','U','V','W','X','Y','Z')

{

"Processing information for OU $targetou"

$DomainRootPath='LDAP://OU='+$targetou+',OU=ORGUSER,DC=contoso,DC=com'

$adsearch = New-Object DirectoryServices.DirectoryAdsearch([adsi]$DomainRootPath)

$adsearch.filter = "(objectclass=user)"

$adsearch.PropertiesToLoad.AddRange(@("name"))

$adsearch.PropertiesToLoad.AddRange(@("lastLogon"))

$adsearch.PropertiesToLoad.AddRange(@("givenName"))

$adsearch.PropertiesToLoad.AddRange(@("SN"))

$adsearch.PropertiesToLoad.AddRange(@("DisplayName"))

$adsearch.PropertiesToLoad.AddRange(@("extensionAttribute1"))

$adsearch.PropertiesToLoad.AddRange(@("extensionAttribute2"))

$adsearch.PropertiesToLoad.AddRange(@("comment"))

$adsearch.PropertiesToLoad.AddRange(@("title"))

$adsearch.PropertiesToLoad.AddRange(@("mail"))

$adsearch.PropertiesToLoad.AddRange(@("userAccountControl"))

$adsearch.Container

$users = $adsearch.findall()

$report = @()

foreach ($objResult in $users)

{

$objItem = $objResult.Properties

$temp = New-Object PSObject

$temp | Add-Member NoteProperty name $($objitem.name)

$temp | Add-Member NoteProperty title $($objitem.title)

$temp | Add-Member NoteProperty mail $($objitem.mail)

$temp | Add-Member NoteProperty displayname $($objitem.displayname)

$temp | Add-Member NoteProperty extensionAttribute1 $($objitem.extensionattribute1)

$temp | Add-Member NoteProperty extensionAttribute2 $($objitem.extensionattribute2)

$temp | Add-Member NoteProperty givenname $($objitem.givenname)

$temp | Add-Member NoteProperty sn $($objitem.sn)

$temp | Add-Member NoteProperty useraccountcontrol $($objitem.useraccountcontrol)

$report += $temp

}

$csvfile="AD-"+$targetou+".csv"

$report | export-csv -notypeinformation $csvfile

"Wrote file for $targetou"

}

----------------------------------------------------------------------------------------------------------------------------------------------------------------

-- Powershell scripts\_sharefolderandpermissions.ps1

# ExportShareInfo.ps1

# This script will export type 0 shares with security info, and provide a hash table of shares

# in which security info could not be found.

#

#reference: http://mow001.blogspot.com/2006/05/powershell-export-shares-and-security.html

#SID was removed from the script. Instead, the username is used to find SID when the import is run

# CHANGE TO SERVER THAT HAS SHARES TO EXPORT

$fileServer = "your file server here"

$date = get-date

$datefile = get-date -uformat '%m-%d-%Y-%H%M%S'

$filename = 'desired destination for csv file here'

#Store shares where security cant be found in this hash table

$problemShares = @{}

Function Get-ShareInfo($shares) {

$arrShareInfo = @()

Foreach ($share in $shares) {

trap{continue;}

write-host $share.name

$strWMI = "\\" + $fileServer + "\root\cimv2:win32\_LogicalShareSecuritySetting.Name='" + $share.name + "'"

$objWMI\_ThisShareSec = $null

$objWMI\_ThisShareSec = [wmi]$strWMI

#In case the WMI query or 'GetSecurityDescriptor' fails, we retry a few times before adding to 'problem shares'

For($i=0;($i -lt 5) -and ($objWMI\_ThisShareSec -eq $null);$i++) {

sleep -milliseconds 200

$objWMI\_ThisShareSec = [wmi]$strWMI

}

$objWMI\_SD = $null

$objWMI\_SD = $objWMI\_ThisShareSec.invokeMethod('GetSecurityDescriptor',$null,$null)

For($j=0;($j -lt 5) -and ($objWMI\_SD -eq $null);$j++) {

sleep -milliseconds 200

$objWMI\_SD = $objWMI\_ThisShareSec.invokeMethod('GetSecurityDescriptor',$null,$null)

}

If($objWMI\_SD -ne $null) {

$arrShareInfo += $objWMI\_SD.Descriptor.DACL | % {

$\_ | select @{e={$share.name};n='Name'},

@{e={$share.Path};n='Path'},

@{e={$share.Description};n='Description'},

AccessMask,

AceFlags,

AceType,

@{e={$\_.trustee.Name};n='User'},

@{e={$\_.trustee.Domain};n='Domain'}

}

}

Else {

$ProblemShares.Add($share.name, "failed to find security info")

}

}

return $arrshareInfo

}

Write-Host "Finding Share Security Information"

# get Shares (Type 0 is "Normal" shares) # can filter on path, etc. with where

$shares = gwmi Win32\_Share -filter 'type=0'

# get the security info from shares, add the objects to an array

Write-Host " Complete" -ForegroundColor green

Write-Host "Preparing Security Info for Export"

$ShareInfo = Get-ShareInfo($shares)

Write-Host " Complete" -ForegroundColor green

Write-Host "Exporting to CSV"

# Export them to CSV

$ShareInfo | select Name,Path,Description,User,Domain,

AccessMask,AceFlags,AceType | export-csv -noType $filename

Write-Host " Complete" -ForegroundColor green

Write-Host "Your file has been saved to $filename"

If ($problemShares.count -ge 1) {

Write-Host "These Shares Failed to Export:"

}

$problemShares

----------------------------------------------------------------------------------------------------------------------------------------------------------------

--

----------------------------------------------------------------------------------------------------------------------------------------------------------------

--