Inventory the directory access rights on file servers

During the migration, it was needed to inventory the directory access on several file servers. So we could easily monitor the access rights on several directories on the file servers. Some applications are started from a UNC path. So we could add the ‘old’ and the ‘new’ group and check if that has been done properly.

For testing I created the following structure:

```
\\DEMOATS-SCCM\DEMO.
\---share1
    +---Appl1
    +---Appl2
    |   \---Test
    +---Appl3
    |   +---Sub1
    |   \---Sub2
    |       \---SubSub1
    \---Appl4
```

I had some challenges:

- Make a difference between inherited and non-inherited rights. I only want to see the differences. But that can be changed with the parameter showInherited
- There are some file servers where all the shares (from the root) have to be inventoried. \\server\share did not work, as the rights where inherited. And the script did not see that properly. So I had to inventory the ‘root’ shares on the server. And go through all the directories. I found the code on StackOverflow.
- And a lot of testing.

The script that I created:

```
<#
.NOTES
================================================================
================================
Created with:     Windows PowerShell ISE
Created on:       03-August-2018
Created by:       Willem-Jan Vroom
Organization:
Filename:         Inventory Permissions on Shares (v02).ps1
================================================================
================================
.DESCRIPTION:

This script writes the directory permissions of the given
shares to a CSV file.

.USAGE:

1.
Run an inventory on the shares
\\server\share,\\server2\share1 and all shares on \\server3\
with the default setting of a search level of 1 and only show
the directories that are not
inherited for the AD group 'Users':
.\"Inventory Permissions on Shares (v02).ps1" -ShareList
\\server\share,\\server2\share1,\\server3\

2.
Run an inventory on the share \\server\share with an search
level of 10 for all the 'Appl'
groups:
.\"Inventory Permissions on Shares (v02).ps1" -ShareList
\\server\share    -NumberOfLevelsToSearch    10    -
GroupNameToSearchFor "Appl"

3.
Run a complete inventory for one server for all groups:
.\"Inventory Permissions on Shares (v02).ps1" -ShareList
\\server\ -NumberOfLevelsToSearch 10 -GroupNameToSearchFor ""
-showInherited
```

```
.VERSION HISTORY:
 v0.1:
   * Initial version.

 v.0.2:
   * Option -Outputfile has been added.
   * Added help text by the options.

 v.0.3:
   * The parameter showInherited has become a switch.

#>

param
(
[Parameter(Mandatory=$true,HelpMessage="Please mention the
shares you want to inventory regarding the permissions. One
name each line.")]
[String[]] $ShareList,

[Parameter(HelpMessage="Give a part of the group name to
search for. Leave empty for all groups. Default = Users")]
$GroupNameToSearchFor    = "Users",

[Parameter(HelpMessage="Give the search level. Default = 1")]
$NumberOfLevelsToSearch  = 1,

[Parameter(HelpMessage="Show inherited directories, if
specified.")]
[switch]$showInherited,

[Parameter(HelpMessage="Mention the output file. Default is
the script name, with csv as the extension.")]
$OutputFile              = ""
)

#
================================================================
================================
# Function block
#
```

```
================================================================
================================

Function Get-NetShares
{

<#
.NOTES
================================================================
================================
Created with:     Windows PowerShell ISE
Created on:       03-August-2018
Created                                                    by:
https://stackoverflow.com/users/2102693/bill-stewart
Organization:
Functionname:     Get-NetShares
================================================================
================================
.DESCRIPTION:

This function finds all the shares that are on a server.

I have found this script here:
https://stackoverflow.com/questions/45089582/using-get-childit
em-at-root-of-unc-path-servername
(C) by https://stackoverflow.com/users/2102693/bill-stewart

#>

param
(
  [String] $ComputerName = "."
)

Add-Type @"
using System;
using System.Runtime.InteropServices;
using System.Text;
[StructLayout(LayoutKind.Sequential,    CharSet    =
CharSet.Unicode)]
public struct SHARE_INFO_1
```

```
{
  [MarshalAs(UnmanagedType.LPWStr)]
  public string shi1_netname;
  public uint shi1_type;
  [MarshalAs(UnmanagedType.LPWStr)]
  public string shi1_remark;
}
public static class NetApi32
{
  [DllImport("netapi32.dll", SetLastError = true)]
  public static extern int NetApiBufferFree(IntPtr Buffer);
   [DllImport("netapi32.dll", CharSet = CharSet.Unicode,
SetLastError = true)]
  public static extern int NetShareEnum(
    StringBuilder servername,
    int level,
    ref IntPtr bufptr,
    uint prefmaxlen,
    ref int entriesread,
    ref int totalentries,
    ref int resume_handle);
}
"@

$pBuffer = [IntPtr]::Zero
$entriesRead = $totalEntries = $resumeHandle = 0
$result = [NetApi32]::NetShareEnum(
  $ComputerName,        # servername
  1,                    # level
  [Ref] $pBuffer,       # bufptr
  [UInt32]::MaxValue,   # prefmaxlen
  [Ref] $entriesRead,   # entriesread
  [Ref] $totalEntries,  # totalentries
  [Ref] $resumeHandle   # resumehandle
)
if ( ($result -eq 0) -and ($pBuffer -ne [IntPtr]::Zero) -and
($entriesRead -eq $totalEntries) ) {
  $offset = $pBuffer.ToInt64()
  for ( $i = 0; $i -lt $totalEntries; $i++ ) {
    $pEntry = New-Object IntPtr($offset)
                                        $shareInfo         =
```

```
[Runtime.InteropServices.Marshal]::PtrToStructure($pEntry,
[Type] [SHARE_INFO_1])
    $shareInfo
                                          $offset         +=
[Runtime.InteropServices.Marshal]::SizeOf($shareInfo)
  }
  [Void] [NetApi32]::NetApiBufferFree($pBuffer)
}
if ( $result -ne 0 ) {
           Write-Error     -Exception     (New-Object
ComponentModel.Win32Exception($result))
}
}

Function Add-EntryToReport
{

<#
.NOTES
================================================================
================================
Created with:     Windows PowerShell ISE
Created on:       03-August-2018
Created by:       Willem-Jan Vroom
Organization:
Functionname:     Add-EntryToReport
================================================================
================================
.DESCRIPTION:

This function adds an entry to the report. When the shares
have been searched, the report is
exported to a CSV file.

#>

param
 (
  $FolderNameToAdd,
  $ErrorMessage = "",
  $ADGroup      = "",
```

```
  $Permissions  = "",
  $Inherited    = ""
 )
   $Record = [ordered] @{"FolderName" = "";"AD Group" =
"";"Permissions" = "";"Inherited" = "";"Error" = ""}
  $Record."FolderName"  = $FolderNameToAdd
  $Record."Error"       = $ErrorMessage
  $Record."AD Group"    = $ADGroup
  $Record."Permissions" = $Permissions
  $Record."Inherited"   = $Inherited
   $Global:Report += New-Object -TypeName PSObject -Property
$Record
}


Function Search-InTheFolder
{

<#
.NOTES
================================================================
================================
Created with:     Windows PowerShell ISE
Created on:       03-August-2018
Created by:       Willem-Jan Vroom
Organization:
Functionname:     Search-InTheFolder
================================================================
================================
.DESCRIPTION:

This function goes through all the folders in the given
location.

The variable $numLevels gives the number of levels that the
search goes. The less the number,
the quicklier the script is.

#>

param
```

```
(
  $RootOfTheShare,
  $numLevels = 1
)

$valNumberOfDirectories = 0
$valCounterOfDirectores = 0

Try
  {
$FolderPath = Get-ChildItem -Path $RootOfTheShare -Directory -
Recurse -Force -Depth $numLevels -ErrorAction SilentlyContinue
Write-Progress -Activity "Counting the number of folders in
the share." -Id 2 -ParentId 1
Foreach ($Folder in $FolderPath)
     {
      $valNumberOfDirectories++
     }
   Foreach ($Folder in $FolderPath)
     {
      $FolderFullName = $Folder.FullName
      Write-Progress -Activity "Going through all the shares."
-Status "Processing share $FolderFullName." -Id 2 -
PercentComplete ($valCounterOfDirectores /
$valNumberOfDirectories * 100) -ParentId 1
          Get-FolderRights -FolderNameToInvestigate
$Folder.FullName
      $valCounterOfDirectores++
     }
  }
  Catch
  {
   Add-EntryToReport -FolderNameToAdd $RootOfTheShare -
ErrorMessage $_.Exception.Message
  }

}



Function Get-FolderRights
{
```

```
<#
.NOTES
================================================================
================================
Created with:     Windows PowerShell ISE
Created on:       03-August-2018
Created by:       Willem-Jan Vroom
Organization:
Functionname:     Get-FolderRights
================================================================
================================
.DESCRIPTION:

This function puts the access information in an array.
If the parameter GroupNameToSearchFor is empty or "" then all
the groups are shown.

#>
Param($FolderNameToInvestigate)
$GroupNameToSearchFor = $GroupNameToSearchFor.ToLower()
Try
    {
    $Acl = Get-Acl -Path $FolderNameToInvestigate -ErrorAction
SilentlyContinue
    foreach ($Access in $acl.Access)
        {
              $Group=$Access.IdentityReference
              if ($GroupNameToSearchFor.Length -ge 1)
               {
                                                  $Position   =
$Group.ToString().ToLower().IndexOf($GroupNameToSearchFor)
               }
                Else
               {
                $Position = 2
               }
              if($Position -ge 1)
              {
               [bool]$blnIsInheriated = $Access.IsInherited
               if([bool]$showInherited)
                {
```

```
                    Add-EntryToReport -FolderNameToAdd
$FolderNameToInvestigate -ADGroup $Access.IdentityReference -
Permissions    $Access.FileSystemRights    -Inherited
[bool]$blnIsInheriated
                }
                   if((-not[bool]$showInherited) -and (-
not[bool]$blnIsInheriated))
                {
                    Add-EntryToReport -FolderNameToAdd
$FolderNameToInvestigate -ADGroup $Access.IdentityReference -
Permissions    $Access.FileSystemRights    -Inherited
[bool]$blnIsInheriated
                }
              }
        }
        }
        Catch
        {
                  Add-EntryToReport  -FolderNameToAdd
$FolderNameToInvestigate -ErrorMessage $_.Exception.Message
        Continue
        }
}

#
================================================================
================================
# End function block
#
================================================================
================================

#
================================================================
================================
# Define the CSV Export File.
#
================================================================
================================

  $currentPath                        = Split-Path -parent
```

```
$MyInvocation.MyCommand.Definition
    $strCurrentFile                                          =
$MyInvocation.MyCommand.Name
  if($OutputFile.Length -eq 0)
   {
    $strCSVFileName                = $strCurrentFile -Replace
".ps1",".csv"
   }
    else
   {
    if($OutputFile.ToLower().IndexOf(".csv") -eq -1)
    {
     $OutputFile = $OutputFile + ".csv"
    }
    $strCSVFileName = $OutputFile
   }

#
=================================================================
==================================
# Check if the string $strCSVFileName is a path. In that case,
nothing has to be done.
# In case it is not a path, then the current location should
be added.
#
=================================================================
==================================


  if (Split-Path($strCSVFileName))
  {
   $CSVExportFile = $strCSVFileName
  }
   else
  {
   $CSVExportFile = $currentPath + "\" + $strCSVFileName
  }

#
=================================================================
==================================
```

```
# Create the folder as a part of $CSVExportFile if not exists.
#
================================================================
================================

  $PathFromCSVExportFile = Split-Path $CSVExportFile
  if(-not(Test-Path(Split-Path $PathFrom$CSVExportFile)))
  {
   New-Item -Path $PathFromCSVExportFile -ItemType Directory
  }

#
================================================================
================================
# Define variables.
#
================================================================
================================

  $arrShares                        = @($ShareList)
  $Global:Report                    = @()
#
================================================================
================================
# Start the job.
#
================================================================
================================

  Clear-Host
  Try
     {
      Import-Module ActiveDirectory
     }
     Catch
     {
       Write-Host The module ActiveDirectory could not be
loaded.
     Exit 1
     }
  Write-Host (Get-Date).ToString('T') " Starting..."
```

```
#
================================================================
================================
# Deletes the CSV file if exists.
#
================================================================
================================

  If(Test-Path $CSVExportFile)
  {
  Remove-Item $CSVExportFile
  }

#
================================================================
================================
# Go through all the shares as defined in the array $arrShares
#
================================================================
================================
  $valCounter       = 1
  $valNumerOfShares = $arrShares.Count
  ForEach ($shareName in $arrShares)
  {
   Write-Progress -Id 1 -Activity "Going through the shares" -
Status  "Checking  share  $shareName  ($valCounter  of
$valNumerOfShares)"  -PercentComplete  ($valCounter  /
$valNumerOfShares*100)
                                  $LastCharacter          =
$shareName.SubString($shareName.Length-1,1)

                                                          #
================================================================
================================
   # If the last character is not a '\' then it is a regular
share. Then it is simple: call the
   # function 'Search-InTheFolder'
   #
   # If the last character is a '\' then only the servername
is given. So first find all the
    # shares on that server. After that, process all the
```

```
subshares.
                                                                    #
=================================================================
================================
   if ($LastCharacter -ne "\")
   {
    Search-InTheFolder -RootOfTheShare $shareName -numLevels
$NumberOfLevelsToSearch
   }
   Else
   {
     $arrShares = Get-NetShares -ComputerName $shareName
     ForEach($objShare in $arrShares)
     {
                                        $LastCharacter      =
($objShare.shi1_netname).SubString(($objShare.shi1_netname).Le
ngth-1,1)
      if($LastCharacter -ne "$")
      {
                                                                    #
=================================================================
================================
       # Ignore C$, D$, IPC$, NETADMIN$ etc.
       # This means that all the hidden shares are ignored.
                                                                    #
=================================================================
================================
       $ServerShareName = $shareName + $objShare.shi1_netname
            Get-FolderRights  -FolderNameToInvestigate
$ServerShareName
        Search-InTheFolder -RootOfTheShare $ServerShareName -
numLevels $NumberOfLevelsToSearch
      }
     }
   }
   $valCounter++
  }

#
=================================================================
================================
```

```
# Output naar een CSV file
#
================================================================
================================

  $Global:Report | Sort-Object -Property FolderName,"AD Group"
| Export-Csv -path $CSVExportFile -NoTypeInformation -Encoding
ASCII
  Write-Host You can open the file $CSVExportFile now.
  Write-Host (Get-Date).ToString('T')  " Ended..."
```

Any feedback to improve this script is appreciated. You can download the scripts here:

1. Link to Inventory Permissions on Shares (v0.1)
2. Link to Inventory Permissions on Shares (v0.2)
3. Link to Inventory Permissions on Shares (v0.3)