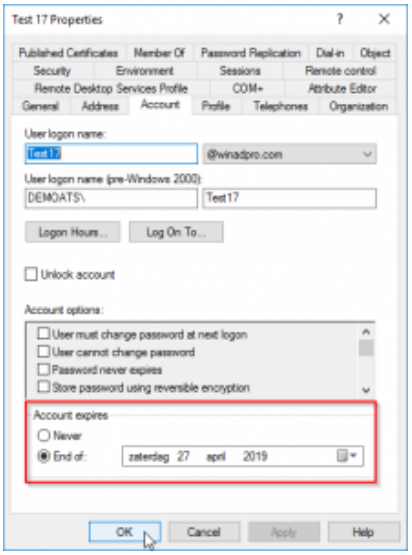
All expired user accounts and the accounts that are about to expire

A colleague of mine asked me to write a script with the Active Directory users with an account about to expire. So I created this script for him:



Account with an expiry date.

```
$arrUsers = @()
$arrOU                                                    =
@("OU=Users,OU=OU1,DC=testdomain,DC=local,DC=lan","OU=Users,OU
=OU2,DC=testdomain,DC=local,DC=lan")

ForEach($objOU in $arrOU)
 {
  $arrUsers += (Search-ADAccount -AccountExpiring -SearchBase
$objOU       -UsersOnly)       |       Select-Object
"AccountExpirationDate","Name",@{Name="OU";Expression={($_."Di
stinguishedName"                                          -split
"=",3)[-1]}},"Enabled","LastLogonDate","LockedOut","ObjectClas
```

```
s","PasswordExpired","PasswordNeverExpires","SamAccountName","
UserPrincipalName"
 }
$arrUsers | Sort-Object -Property "OU","Name" | Export-Csv
"c:\temp\usersabouttoexpire.csv" -NoTypeInformation
```

It does what it should do, but not flexible. It can be used in one situation but needs modification for other situations. Also, you cannot specify the number of days when the account is about to expire.

So I created the script AccountsAboutToExpire (v10).ps1.

There are some parameters:

- **OUs**: Specify the OUs, seperated by a comma.
- **IncludeChildOUs**: Use this switch to include the child OU's.
- **NumberOfDaysToExpirationDate**: Specify the number of days in which the account expires. Default = 7

This script does not only give the accounts that are about to expire, but also the expired user accounts.

A demonstration of this script can be found on my YouTube channel: the script and AppV Repository or view it below:

The script:

```
<#
.SYNOPSIS
    Gives the accounts that about to expire within the given
period.

.DESCRIPTION
    This script gives the accounts that are about to expire
within the given period in a CSV file.

.EXAMPLE
    Reports all the accounts that are about to expire in the
OUs   OU=Users,OU=OU1,DC=testdomain,DC=local,DC=lan   and
```

```
OU=Users,OU=OU2,DC=testdomain,DC=local,DC=lan
          ."\AccountsAboutToExpire  V10).ps1"  -OUs
"OU=Users,OU=OU1,DC=testdomain,DC=local,DC=lan","OU=Users,OU=O
U2,DC=testdomain,DC=local,DC=lan"

.EXAMPLE
    Reports all the accounts that are about to expire in the
OU DC=testdomain,DC=local,DC=lan, including the child OU's
          ."\AccountsAboutToExpire  V10).ps1"  -OUs
"DC=testdomain,DC=local,DC=lan" -IncludeChildOUs

.EXAMPLE
    Reports all the accounts that are about to expire witin 60
days in the OU DC=testdomain,DC=local,DC=lan, including the
child OU's
          ."\AccountsAboutToExpire  V10).ps1"  -OUs
"DC=testdomain,DC=local,DC=lan"  -IncludeChildOUs  -
NumberOfDaysToExpirationDate 60

.NOTES
    Author:  Willem-Jan Vroom
    Website:
    Twitter: @TheStingPilot

v0.1:
   * Initial version.

v1.0:
   * Included:
      - All the expired user accounts
      - Sort on expiration date.
#>

[CmdLetBinding()]

param
(
# Specify the OUs, seperated by a comma.
[Parameter(Mandatory=$True)]
[string[]] $OUs,
```

```
# Use this switch to include the child OU's.
[Parameter(Mandatory=$False)]
[switch]  $IncludeChildOUs,

# Specify the number of days in which the account expires.
Default = 7
[Parameter(Mandatory=$False)]
[string]  $NumberOfDaysToExpirationDate=7
)

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

  Function Get-AccountsAboutToExpire
   {
    param
     (
      [string] $OU,
      [Switch] $InclChildOUs,
      [string] $NumDays
     )

    $arrItems       = @()
    $arrAboutToExp  = New-Object PSObject
    $arrExpired     = New-Object PSObject
    $strSearchScope = "OneLevel"

    if($InclChildOUs)
     {
      $strSearchScope = "SubTree"
     }

    Try
     {
```

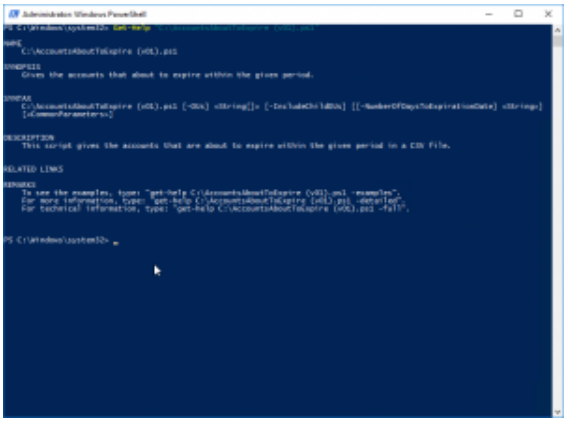
```
      $arrAboutToExp  = (Search-ADAccount -AccountExpiring -
SearchBase $objOU -UsersOnly -SearchScope $strSearchScope -
TimeSpan       $NumDays)       |       Select-Object
"AccountExpirationDate","Name",@{Name="OU";Expression={($_."Di
stinguishedName"                                          -split
"=",3)[-1]}},@{Name="ExpiredAccount";Expression={$False}},"Ena
bled","LastLogonDate","LockedOut","PasswordExpired","PasswordN
everExpires"
      $arrExpired     = (Search-ADAccount -AccountExpired  -
SearchBase $objOU -UsersOnly -SearchScope $strSearchScope)
|                                                 Select-Object
"AccountExpirationDate","Name",@{Name="OU";Expression={($_."Di
stinguishedName"                                          -split
"=",3)[-1]}},@{Name="ExpiredAccount";Expression={$True}},
"Enabled","LastLogonDate","LockedOut","PasswordExpired","Passw
ordNeverExpires"
      $arrItems      += $arrAboutToExp
      $arrItems      += $arrExpired
     }
      Catch
     {
      Write-Warning "The OU $OU has not been found."
     }

    Return $arrItems

   }

Function Check-HasAdminRights
 {

  <#
  .NOTES
================================================================
============================================================
  Created with:     Windows PowerShell ISE
  Created on:       11-January-2019
  Created by:       Willem-Jan Vroom
  Organization:
  Functionname:     Check-HasAdminRights
================================================================
```

```
================================================================
  .SYNOPSIS

   This function checks if an user has admin rights. The
function returns $true or $false

  #>

        If    (([Security.Principal.WindowsPrincipal]
[Security.Principal.WindowsIdentity]::GetCurrent()).IsInRole([
Security.Principal.WindowsBuiltInRole] "Administrator"))
   {
    Return $True
   }
    else
   {
    Return $False
   }
 }

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

#
================================================================
============================================================
# Stop the script for a non admin user
#
================================================================
============================================================

  if(-not(Check-HasAdminRights))
  {
   Write-Error "The current user has no admin rights. Please
```

```
rerun the script with elevated rights." -Category
PermissionDenied
   Exit 999
  }

#
================================================================
================================================================
================
# Check if the module ActiveDirectory has been loaded.
#
================================================================
================================================================
================

  if(-not(Get-Module -ListAvailable ActiveDirectory))
   {
    Write-Warning "The module ActiveDirectory is not found.
Thus quitting."
    Exit 9
   }

#
================================================================
================================================================
================
# Define the variables
#
================================================================
================================================================
================

  $arrUsers      = @()
        $strCurrentDir    =    Split-Path    -parent
$MyInvocation.MyCommand.Definition
#
================================================================
================================================================
================
# Define the output file
#
```

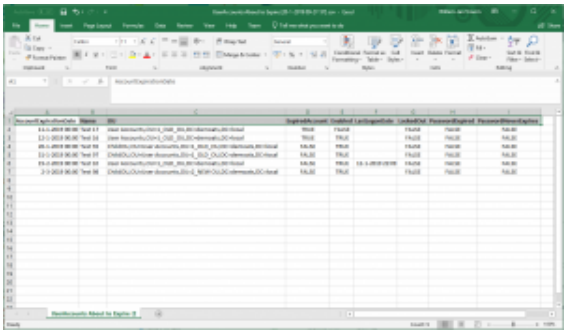
```
=================================================================
=================================================================
================

  $strOutputFile = "UserAccounts About to Expire (" + (Get-
Date).ToString('G') + ").csv"
     $strOutputFile  =  $strOutputFile.replace(":","-
").Replace("/","-")
  $strOutputFile = $strCurrentDir + "\" + $strOutputFile

#
=================================================================
=================================================================
================
# Process the OUs.
#
=================================================================
=================================================================
================

  ForEach($objOU in $OUs)
   {
    if($IncludeChildOUs)
     {
       $arrUsers += Get-AccountsAboutToExpire -OU $objOU -
NumDays $NumberOfDaysToExpirationDate -InclChildOUs
     }
      else
     {
       $arrUsers += Get-AccountsAboutToExpire -OU $objOU -
NumDays $NumberOfDaysToExpirationDate
     }
   }

#
=================================================================
=================================================================
================
# Write the output to a csv file.
#
=================================================================
```

```
================================================================================
        $arrUsers | Sort-Object -Property "AccountExpirationDate","OU","Name" | Export-Csv $strOutputFile -NoTypeInformation
```



Get default help.

Get detailed help.



Output in Excel.

Current version: AccountsAboutToExpire (v10)

Previous versions:

- AccountsAboutToExpire (v01)